UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

David Moličnik

IZDELAVA SPLETNEGA MODULA ZA POROČNAJE O ŠTIPENDIJAH S POMOČJO OGRODJA VAADIN

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Ljubljana, 2016

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

David Moličnik

IZDELAVA SPLETNEGA MODULA ZA POROČNAJE O ŠTIPENDIJAH S POMOČJO OGRODJA VAADIN

DIPLOMSKO DELO NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

MENTOR: prof. dr. Viljan Mahnič

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Predstavite informacijski sistem centrov za socialno delo in umestitev modula za poročanje o štipendijah v njegov kontekst. Podrobno opišite zahteve, ki jih mora izpolnjevati ta modul, in njegovo realizacijo. Pri tem posvetite posebno pozornost izbiri ustreznega ogrodja, opišite izbrano ogrodje in njegove značilnosti, predstavite implementcijo zahtevane rešitve in prikažite njeno uporabo v praksi.

ZAHVALA

Zahvaljujem se mentorju prof. dr. Viljanu Mahniču za pomoč in vodenje pri opravljanju diplomskega dela. Zahvala gre tudi moji družini, ki me je tekom študija tako ali drugače podpirala.

Kazalo

Seznam uporabljenih kratic

CSD

Center za socialno delo

IS CSD

Informacijski sistem Centrov za socialno delo

ISCSD

IS Centrov za socialno delo za potrebe odločanja po ZSDP, ZŠtip…

ISCSD2

IS Centrov za socialno delo za potrebe odločanja ZUPJS, …

ZŠtip-1

Zakon o štipendiranju (ZŠtip-1), Ur. L. RS 56/2013

ZŠtip

Zakon o štipendiranju, predhodnik zakona ZŠtip-1

ZSDP

Zakon o starševskem varstvu in družinskih prejemkih

Sklad

Javni sklad RS za razvoj kadrov in štipendij (tudi Sklad)

MDDSZ, Ministrstvo

Ministrstvo za delo, družino, socialne zadeve in enake možnosti

SURS

Statistični urad RS

RIA

»Rich Internet Application« bogate oz. obogatene internetne aplikacije

CRP

Centralni register prebivalstva oz. spletna storitev, ki nudi podatke o CRP

MIZŠ

Ministrstvo za izobraževanje, znanost in šport, tudi z njo povezana spletna storitev

AJPES

Agencija republike Slovenije za javnopravne evidence in storitve oz. z njo povezana spletna storitev

GWT

Google Web Toolkit – ogrodje za razvoj bogath internetnih aplikacij.

JS

JavaScript – skriptni jezik, ki teče v spletnih brskalnikih.

IDE

Integated Development Environment – integrirano razvijalsko okolje.

API

(Application Programming Interface) Vmesnik za programiranje aplikacij – to je vmesnik preko katerega lahko razvijalci uporabijo in koristijo določen sklop funkcionalnosti (največkrat zapakiranih v obiki knjižnice).

HTTP

HyperText Transfer Protocol – glavni protokol za prenos informacij prek spleta.

UI

uporabniški vmesnik (User Interface) – se uporablja tako za splošni pojem uporabniškega vmesnika kot tehnični pojem UI razred (UI class).

CSS

Cascading Style Sheets – kaskadne predloge v obliki slogovnega jezika za prikaz spletnih strani.

Sass

Syntactically Awesome Style Sheets – razširitev CSS predlog.

DBMS

Data Base Management System – sistem za upravljanje s podatkovnimi zbirkami.

JPA

Java Persistence API – rešitev, ki omogoči obstojno preslikovanje med objekti / entitetami in podatkovno zbirko.

WTP

Web Tools Platform – to je razširitev za za razvojno okolje Eclipse z orodji za razvoj spletnih in Java EE aplikacij.

Povzetek

Naslov: IZDELAVA SPELTNEGA MODULA ZA POROČANJE O ŠTIPENDIJAH S POMOČJO OGRODJA VAADIN

Diplomsko delo govori o ogrodju za razvoj bogatih spletnih aplikacij Vaadin; kako je ogrodje sestavljeno, kaj so njegove prednosti in za kakšen tip aplikacij je ogrodje najbolj primerno. Predstavljen je tudi konkreten problem in obstoječ sistem v katerega je bila umeščena končna rešitev. Opredeljeno je tudi razvojno okolje, s katerim smo implementirali rešitev. Na koncu tega dela je predstavljena konkretna uporaba tega ogrodja na primeru izdelave spletnega modula za poročanje o štipendijah.

Ključne besede: spletne aplikacije, RIA, Java, okolje Java EE, ogrodje Vaadin

Abstract

Title: IMPLEMENTATION OF AN ONLINE REPORTING MODULE ABOUT SCHOLARSHIPS USING THE VAADIN FRAMEWORK

In this thesis we will talk about framework for developing Rich Internet Applications called Vaadin. Will look at how the framework is assembled, what are its strengths and what type of application this framework is most suitable for. We will also present a specific problem and the existing system upon which the final solution is built. We'll define a development environment in which we implemented the solution. At the end we'll present a concrete application of framework on the case of making online reporting module for scholarships.

.

Keywords: Web Applications, RIA, Java, Java EE, Vaadin Framework

1. Uvod

Štipendije so opredeljene kot instrument za spodbujanje oziroma krepitev razvojne politike, za spodbujanje in povečevanje izobrazbene ravni prebivalstva ter odpravo strukturnega neskladja na trgu dela in posredno doseganje večje zaposlenosti. Štipendije mladim omogočajo pridobitev želene izobrazbe. Stopnja izobraženosti v državi se s tem povečuje, gospodarstvo pa dobi usposobljeno delovno silo. Zakon, ki ureja to področje, se imenuje Zakon o štipendiranju ZŠtip-1, ki ga je sprejel državni zbor Republike Slovenije na seji dne 20. junija 2013.

Osrednja tema diplomskega dela je informatizacija tega zakona v delu, ki se nanaša na nadzor in poročanje, s predstavitvijo uporabljenih tehnologij pri implementaciji rešitve. Na konkretnem vsebinskem problemu bomo z upoštevanjem obstoječe infrastrukture opredelili izbiro razvojnega ogrodja (Vaadin) in predstavili spletni modul s strani izdelave uporabniškega vmesnika, ki je glede na svoj obseg (število uporabnikov, zahtevnost) tipičen primer uporabe tega ogrodja v poslovnih aplikacijah.

Kot razvijalci imamo danes na razpolago celo paleto ogrodij za razvoj spletnih aplikacij, tako odprtokodnih kot lastniških, ki se med sabo razlikujejo v različnih pogledih. Za njihovo uporabo se odločamo na podlagi različnih kriterijev [1]:

- *Namen ogrodja in kontekst uporabe* – vsako ogrodje je narejeno z določenim ciljem in odgovor na vprašanje, ali ogrodje zadostuje našim zahtevam, je eden od temeljnih kriterijev, s katerim upravičimo njegovo uporabo.
- *Varnost* – načeloma mora vsaka aplikacija poskrbeti za varnost in zmanjšati izpostavljenost, stopnja varnosti pa je odvisna od začetnih zahtev in od konteksta uporabe.
- *Trajnost* – pri izbiri ogrodja moramo upoštevati in preučiti možnosti, da bo ogrodje izpolnilo tudi vse morebitne razširitve in nadgradnje , ki se lahko pojavijo v prihodnje,
- *Popularnost in velikost skupnosti, ki uporablja to ogrodje* – bolj je ogrodje razširjeno in prepoznano v skupnosti razvijalcev, bolj se bo ogrodje razvijalo v prihodnje (vtičniki, nivo kvalitete, novosti). Upoštevati je potrebno tudi trend uporabe (se zvišuje / upada).
- *Podpora*: Preučiti moramo, kako hitro in s kakšnim naporom pridemo do odgovorov, ki se pojavijo v zvezi z uporabo ogrodja. Pri večjih ogrodjih moramo upoštevati tudi možnost izobraževanja in pridobivanja certifikatov.

- *Licenc*a: Pod kakšno licenco je ogrodje, ki ga nameravamo uporabljati kasneje, vpliva tudi na to pod kakšno licenco bo naša izvorna koda.(GPL LGPL, Apache Licence, MIT)
- *Znanje in veščine*: Če ogrodja še ne poznamo moramo upoštevati tudi čas za učenje (učna krivulja) in koliko nam bo obstoječe znanje pri tem pomagalo. Mogoče bomo rabili tudi pomoč od zunaj – kaj nam ponuja trg.
- *Dokumentacija in literatura*: Dobra dokumentacija in kakovost literature nam zelo olajšata uporabo in sta lahko odločilnega pomena pri odločitvi.

Pri izbiri ogrodja za izdelavo spletnega uporabniškega vmesnika se poleg prej omenjenih kriterijev odločamo tudi na podlagi dejstva, ali je ogrodje bolj odjemalsko ali bolj strežniško orientirano.

Pri izbiri orodja za sam razvoj aplikacije pa smo se omejili na razvojna orodja, ki so odprtokodna in prosto dostopna širši javnosti.

V prvem delu tega diplomskega dela si bomo ogledali sistem, v katerega bo rešitev postavljena. Nato si bomo ogledali orodja za razvoj, samo ogrodje, njegove prednosti in razloge za izbor le tega. Na koncu bomo predstavili še delovanje spletnega modula. Končni rezultat diplomskega dela je delujoča spletna aplikacija, ki omogoča vsem vpletenim subjektom izpolnjevanje svojih nalog določenih z zakonom.

2. Obstoječe okolje in opis problema

Odločanje o upravičenosti in izplačevanje transferjev upravičencem na podlagi pravic, ki jih določa zakonodaja na področju, za katerega je pristojno ministrstvo, je ena od poglavitnih nalog Ministrstva za delo, družino, socialne zadeve in enake možnosti (v nadaljevanju MDDSZ). Za potrebe učinkovitega izvajanja temeljnih nalog in področne zakonodaje MDDSZ in centrov za socialno delo (v nadaljevanju CSD), je bil vzpostavljen enoten informacijski sistem centrov za socialno delo (IS CSD).

IS CSD je sestavljen iz dveh ločenih zaokroženih sistemov - ISCSD in ISCSD2, ki jih sestavlja več , med sabo prepletenih, modulov.

IS CSD je v uporabi na CSD-jih, MDDSZ, v CE (centralna enota pri CSD Ljubljana Bežigrad) in na Javnemu skladu Republike Slovenije za razvoj kadrov in štipendij

Glede na naravo in dinamiko dela se ocenjuje, da trenutno informacijski sistem naenkrat uporablja največ 400 uporabnikov.

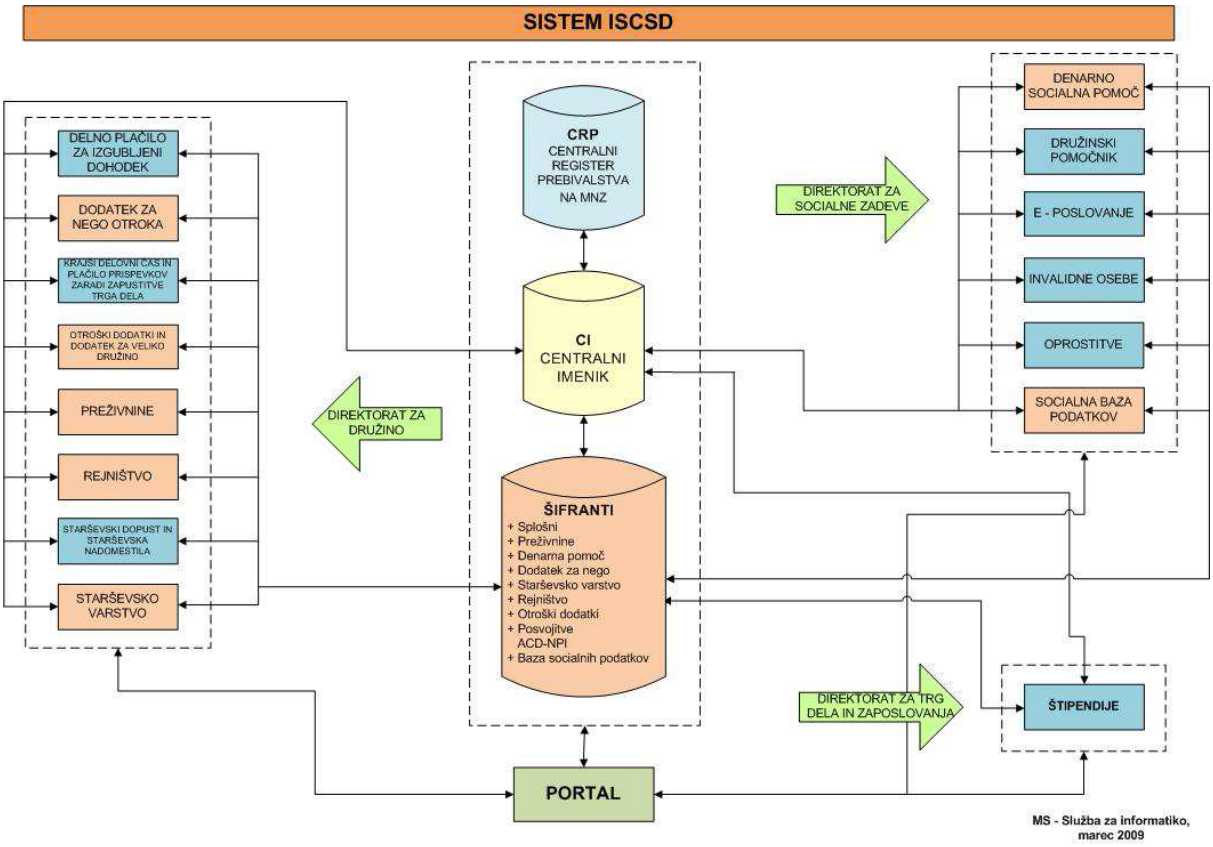
2.1 Opis sistema ISCSD

ISCSD je sistem, ki je v uporabi od leta 2000 in pretežno pokriva področje zakonodaje družinskih prejemkov in starševskega varstva ter štipendij in tako informacijsko podpira izvajanje aktivnosti na vseh ključnih področjih dela centrov za socialno delo[3].

ISCSD je trenutno sestavljen iz sedemnajstih modulov, ki sta jih na podlagi več ločenih javnih naročil razvila dva zunanja izvajalca, ki trenutno vzdržujeta različno število sklopov, eden od izvajalcev pa vzdržuje tudi skupne objekte, kamor sodijo skupni šifranti, centralni imenik in dosjeji. Dostop do modulov je glede na pristojnost razdeljen na posamezne organe (Slika 2-1) [3]:

- starševsko varstvo,
- šifranti,
- preživnine,
- dodatek za nego otroka,
- rejništvo,
- posvojitve,
- delno plačilo za izgubljeni dohodek,
- družinski pomočnik,
- krajši delovni čas,
- pravice do starševskega dopusta, nadomestila in ostale pravice,
- oprostitve plačila socialnovarstvenih storitev,
- baza socialnih podatkov,
- zoisove štipendije,
- kadrovske štipendije,
- futura štipendije,

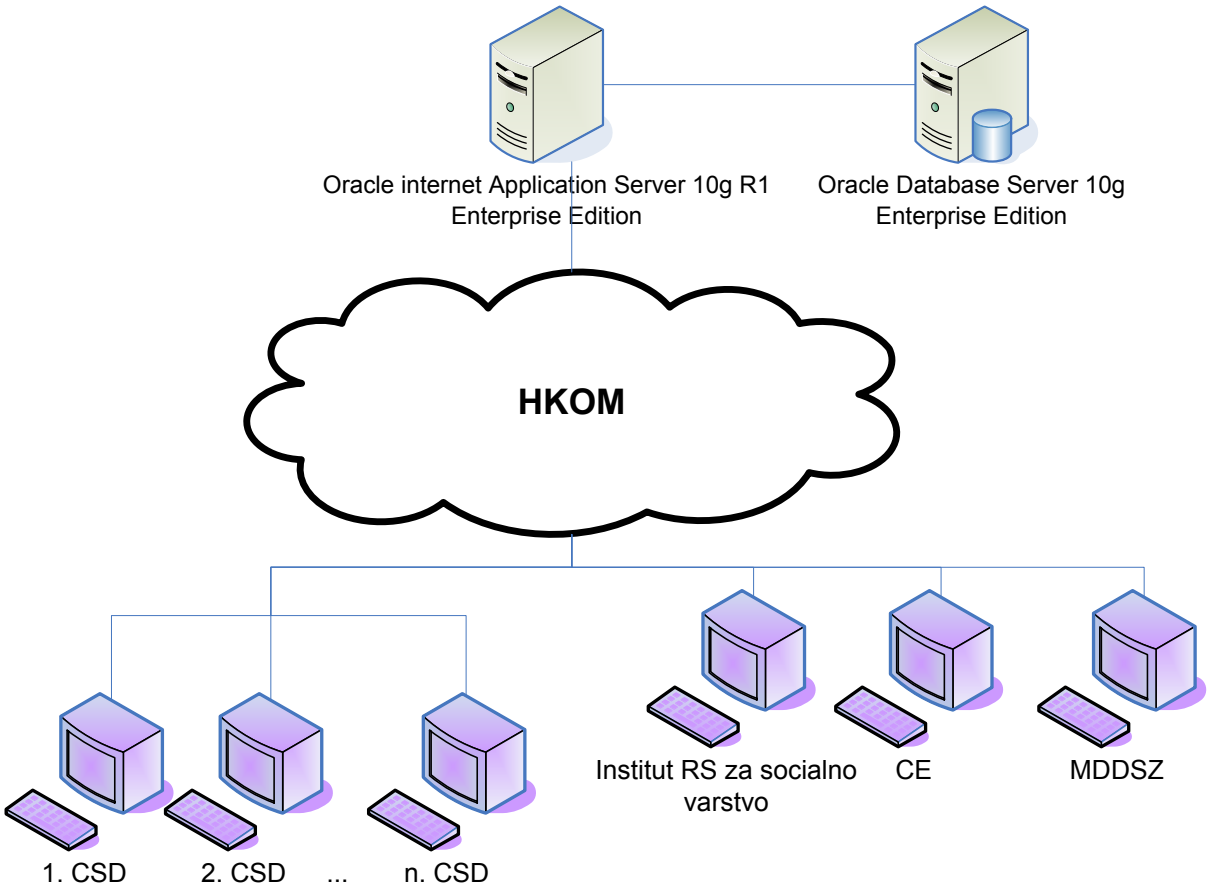
- posebni programi,
- evropska uredba.



Slika 2-1: grafični prikaz modulov v ISCSD in porazdelitev glede na pristojnost organov

2.1.1 Arhitektura ISCSD in omrežje

Arhitektura vključuje centralni podatkovni strežnik, na katerega so preko omrežja HKOM priključene delovne postaje na CSD, MDDSZ, CE... (Slika 2-2Slika 2-1). MDDZS in CSD so priključeni na HKOM (Hitro komunikacijsko omrežje RS, Ministrstva za javno upravo – Direktorata za e-upravo in upravne procese) večinoma že z optično povezavo[3].



Slika 2-2: prikaz privatnega omrežja HKOM

Prostrano omrežje državnih organov HKOM je privatno omrežje, ki je zasnovano za prenos podatkov med posameznimi zaključenimi celotami (CURS, DURS, …) in med posameznimi končnimi uporabniki in centralnim sistemom aplikativnih in podatkovnih strežnikov in storitev (elektronska pošta, Internet, klicni dostopi, …) [2].

2.1.2 Razvojno okolje obstoječega sistema

ISCSD je v celoti razvit v programskem okolju *Oracle*. Za bazo podatkov se uporablja *Oracle Database*. Uporabniški vmesnik je narejen v *Oracle Forms*. Za izpise se uporablja orodje *Oracle Reports* [3].

2.2 Opis sistema ISCSD2

V letu 2011 je ministrstvo na podlagi Zakona o uveljavljanju pravic iz javnih sredstev (Uradni list RS, št. 62/10 in 40/11) dalo v izdelavo nov informacijski sistem (ISCSD2), ki pokriva 4 pravice in 10 subvencij oziroma plačil.

ISCSD2 je osnovno orodje pri odločanju o pravicah iz javnih sredstev s področja socialnovarstvenih prejemkov, družinskih prejemkov in štipendiranja, njegovo brezhibno delovanje pa predpogoj za odločanje o teh pravicah (denarna socialna pomoč, varstveni dodatek, otroški dodatek, državna štipendija ter pripadajoča plačila in subvencije).

S sprejetjem ZUPJS v letu 2012 so se iz ISCSD v ISCSD2 prenesli določeni sklopi/pravice, v okviru sklopa Štipendij so se zaradi področne zakonodaje v ISCSD2 razvile Državne štipendije, v ISCSD pa so ostale delujoče ostale štipendije, ki so v domeni Sklad-a. Sistem je ostal v obliki kot je bil razvit v letu 2008 z implementacijo nekaterih dodatnih funkcionalnosti ter povezljivostjo z ISCSD2.

Tako sedaj sistem nudi podporo odločanju za:

- otroški dodatek,
- denarne socialne pomoči,
- varstveni dodatek,
- štipendije,
- znižano plačilo vrtca,
- subvencije malice za učence in dijake,
- subvencije za kosila za učence,
- oprostitev plačila socialnovarstvenih storitev,
- prispevek k plačilu družinskega pomočnika,
- subvencije najemnine,
- pravico do kritja razlike do polne vrednosti zdravstvenih storitev,
- pravico do plačila prispevka za obvezno zdravstveno zavarovanje,

- pravico do plačila pogrebnine,
- pravico do plačila posmrtnine.

2.3 Ocena velikosti sistema

Izplačevanje, evidentiranje in obdelava podatkov v IS CSD potekajo na podlagi 12 različnih zakonov in tvori izredno kompleksno okolje. Letno se preko sistema IS CSD izplača upravičencem nekaj manj kot 1 milijarda EUR. IS CSD je eden največjih sistemov v državi, zagotovo pa glede na vsebino izredno občutljiv (socialni transferji). IS CSD gostuje na centralni informacijski infrastrukturi ministrstva za javno upravo. IS CSD deluje že od leta 2000 in ga trenutno uporablja več kot 2200 (Tabela 2-1) različnih uporabnikov za različne namene, kot je to razvidno iz spodnje preglednice.

Št. uporabnikov	Organizacija	Namen
1600	62 CSD-jev, Sklad za razvoj kadrov in štipendiranje	odločanje o pravicah
100	MDDSZ	izplačevanje, nadzor, analize, podatki, reševanje pritožb
2	Skupnost CSD, Inštitut Republike Slovenije za socialno varstvo	statistike (analize)
500	Občine, Upravne enote, MIZŠ	vpogled v pravice
30	Izvajalci IS CSD	zagotavljanje delovanja sistema

Tabela 2-1: število uporabnikov glede na organizacijo in namen uporabe IS CSD

2.4 Zakonska podlaga za razvoj novega modula

Zakon ZŠtip-1 je bil objavljen v Uradni list RS, št. 56/2013 z dne 2. 7. 2013 in se je začel uporabljati 1. januarja 2014. Nadomestil je dotedanji zakon ZŠtip in s svojo veljavnostjo

prenehal veljavnost posameznih podzakonskih aktov (pravilniki o dodeljevanju posameznih vrst štipendij).

Glavni razlog in zakonsko podlago za razvoj novega modula za poročanje podaja ZŠtip-1 v IX. poglavju (110. do 113. člena - NADZOR IN POROČANJE), kjer so opredeljene dolžnosti in naloge Ministrstva in Sklada. Prav tako ZŠtip-1 dolžnosti poročanja nalaga drugim podeljevalcem štipendij. Nadzor nad izvrševanjem določb zakona, glede štipendij, ki jih podeljuje Sklad, izvaja Ministrstvo. Sklad mora po potrebi, najmanj pa enkrat letno, Ministrstvu poročati o vrsti, številu in višini podeljenih štipendij. Delodajalci morajo Skladu sporočiti potrebe v zvezi s podelitvijo kadrovskih štipendij najkasneje do konca januarja za naslednje šolsko/študijsko leto (za namen informiranja kandidatov).

O podeljenih štipendijah morajo Skladu letno, do konca koledarskega leta, za tekoče šolsko/študijsko leto poročati tudi subjekti iz 4. člena ZŠtip-1, ki štipendije podeljujejo po drugih predpisih. Zakon določa, da Sklad obrazec za poročanje podatkov objavi na svoji spletni strani. Enkrat letno mora Ministrstvo na zahtevo SURS-u poročati o vseh dodeljenih štipendijah v preteklem letu.

2.5 Opis zahtev

Subjekti iz 4. člena ZŠtip-1, ki so dolžni poročati Skladu o dodeljenih štipendijah, so:

- upravne enote (pristojne za vojne invalide, štipendije otrokom padlih v vojni za Slovenijo 1991): Dodelujejo se na podlagi zakona o posebnih pravicah žrtev v vojni za Slovenijo 1991 (ZPPZV91),
- vlada RS (kadrovske štipendije v organih državne uprave, štipendije Vlade RS, ki štipendije razpisuje za potrebe kadrovanja za posamezni državni organ),
- različna ministrstva (namenske štipendije, štipendije različnih ministrstev),
- slovenska vojska (štipendije v Slovenski vojski),
- tuje vlade (štipendije na podlagi mednarodnih sporazumov, razen če štipendije na podlagi mednarodnih sporazumov dodeljuje sklad),
- posamezni delodajalci (druge štipendije in prejemki, ki so po namenu in načinu vračanja primerljivi štipendijam). Dodeljujejo jih posamezni delodajalci na podlagi internih predpisov,

- poročati morajo tudi vsi ostali ki jih zakon ne zajema, npr. občine, zasebne fundacije in drugi.

Vsi ti subjekti morajo Skladu poročati o podatkih iz 1. – 3., 8. – 12., 15., 19. alineje 2. odstavka 105. člena tega ZŠtip-1. Podatki, o katerih morajo poročati Skladu so:

- (1.) osebno ime,
- (2.) EMŠO,
- (3.) naslov, občina (šifra občine) stalnega oziroma začasnega prebivališča,
- (8.) naziv izobraževalne ustanove,
- (9.) ime izobraževalnega programa in smeri,
- (10.) vrsta in področje izobraževanja,
- (11.) identifikacijska številka izobraževalnega programa,
- (12.) letnik izobraževanja,
- (15.) status štipendista (dijak, študent, udeleženec izobraževanja odraslih),
- (19.) firma ali naziv, matična in davčna številka štipenditorja.

Vsem tem subjektom mora sistem omogočiti vnos naštetih podatkov. Te zahteve izhajajo neposredno iz zakona.

Kot prvotne zahteve je naročnik modula navedel naslednje:

- dostopnost modula zunaj HKOM omrežja (preko spletne strani Sklada),
- možnost poročanja o štipendijah tretjih subjektov na podlagi pooblastila,
- uporaba odprtokodnih in prosto-dostopnih orodij,
- sporočanja preko e-mail strežnika,
- za potrebe preverjanja in boljše kvalitete zbiranja podatkov se modul poveže z različnimi spletnimi servisi; povezava s CRP za potrebe preverjanje podatkov o študentih (imena, priimka in občine stalnega bivanja), povezava z MIZŠ za potrebe

preverjanja podatkov o šolanju (zavod, program, letnik in status) in povezava z AJPES za potrebe preverjanja podatkov o podjetjih (davčna in matično številka),

- dostop do ISCSD šifrantov:
  - šifrant zavodov,
  - šifrant programov,
  - šifrant letnikov,
  - šifranta vrst in področij izobraževanja,
  - tabela obvestil,
  - šifrant občin,
- več nivojev uporabnikov (vsaj trije nivoji),
- avtentičnost in avtorizacija uporabnikov,
- uporaba obstoječe infrastrukture (in postavljenih sistemov v čim večji meri z namenom zniževanja stroškov izvedbe novega modula),

Hkrati pa je naročnik izrazil željo, da naj bo modul izveden kot namizna aplikacija oz. naj bo uporabniška izkušnja modula podobna namizni (*desktop*) aplikaciji v primeru izvedbe modula kot spletne aplikacije.

2.6 Informacijska rešitev

Trenutno ISCSD vsebuje podatke o štipendijah, ki jih dodeljuje Sklad, v ISCSD2 pa so na voljo podatki o državnih štipendijah. Sistem ISCSD ne omogoča pregleda nad dodeljenimi štipendijami, ki jih dodeljujejo subjekti iz 4. člena ZŠtip-1; ti podatki se po vedenju izvajalca, do vpeljave novega modula, niso zbirali nikjer.

Izvedba novega modula, kot del »zaprtih sistemov« ISCSD ali SICSD2 ni prišlo v poštev (zaradi velikega števila zunanjih uporabnikov), ker bi to pomenilo prevelik poseg v obstoječe delovanje in spremenilo prvotni namen obeh sistemov. Hkrati pa drži to, da je v obstoječih rešitvah razvito že veliko funkcionalnosti, ki bi zelo skrajšale in poenostavile (beri pocenile) implementacijo novega modula.

Kot je znano, se v zadnjih nekaj letih pojavlja veliko število poslovnih aplikacij v obliki spletnih aplikacij oz. bogatih spletnih aplikacij (RIA). Tako odpade vsakršno nameščanje in posodabljanje verzije za različne platforme. Uporabnik rabi samo spletni brskalnik in dostop do interneta. Zato nam ni potrebno posebej navajati razloge, da smo se odločili prav za to obliko izvedbe modula (namesto npr. namizne aplikacije).

3. Izbira ogrodja

Iz prejšnjega poglavja je razvidno, da bo implementacija modula temeljila na že postavljeni strežniški infrastrukturi (*JBoss*), kar nam nekako narekuje uporabo javanskega ogrodja, ki je bolj strežniško orientirano *(server-side framewor*k). Z upoštevanjem zahteve, da se naj pri razvoju modula uporabljajo odprtokodna orodja, se izbira omeji na nekaj ogrodij, ki bi lahko ustrezala tem zahtevam.

Kot smo že prej omenili je naročnik izrazil tudi željo, da naj bi spletna aplikacija čim bolj posnemala namizne aplikacije po uporabniški izkušnji. Tako smo pod drobnogled postavili dve javanski ogrodij za izdelavo spletnih aplikacij GWT (*Google Web Toolkit*) in *Vaadin*.

V nadaljevanju bomo skušali ugotoviti prednosti in slabosti posameznega ogrodja. Najprej bomo na kratko opisali oba ogrodja, nato primerjali med sabo in na koncu podali še končno odločitev in razlog za to.

3.1 Ogrodje GWT

Ogrodje GWT je razvojno orodje za izdelavo bogatih spletnih aplikacij oz. RIA. Z njim lahko optimiziramo kompleksne AJAX aplikacije, ki tečejo v brskalniku. Razvijalcu omogoča razvijanje RI aplikacij v Javi, pri tem pa se razvijalcu ni potrebno spraševati o kompatibilnosti razvite kode med različnimi brskalniki . GWT je razvilo podjetje Google in je trenutno pod *Apache License ver. 2.0*[5].

Glavne lastnosti ogrodja so [6]:

- dinamične UI gradnike narejene za ponovno uporabo; razvijalci uporabijo prej definirane razrede, ki implementirajo določene dimanične UI lastnosti (kot so povleci in spusti, drevesni gradniki, gumbi itd.) in s tem pridobijo veliko časa,
- preprost RPC mehanizem,
- upravljanje z zgodovino v brskalniku *Browser history management*),
- podporo polnega Java razhroščevanja,

- GWT sam poskrbi za nekatere razlike med browserji (v smislu intrerpretiranja *JavaScript*-a in *DOM*-a),
- integracija *Unit* testiranja,
- podpora internacionalizaciji in lokalizaciji,
- HTML podpora za vsebnik (*canvas*),
- možnost kombiniranja laste JS kode v java kodi z uporabo JSNI (*JavaScript Native Interface*),
- *open source*,
- načrtovanje in razvoj na tipičen OO način (*object oriented*),
- odkrivanje vseh *compiler-time* napak (napake v času prevajanja kode) kot so napačni tipi in sintaktične napake (*typos*),
- podpira druge API-je v lasti podjetja *Google*,
- možnost *obfuskacije* kode, s čemer se zmanjša tudi velikost JS kode za prenos,
- lepo število knjižnic razvitih od podjetja *Google* in drugih strank, ki razširjajo funkcionalnost GWT-ja,
- širok krog razvijalcev.

Kot glavna komponento GWT orodja bi izpostavil GWT prevajalnik, ki java kodo prevede v *JavaScript*, ki se izvaja na strani odjemalca.

Aplikacije razvite z GWT lahko tečejo v dveh načinih [5]:

- razvijalski način (*development mode* oz. *hosted mode*), kjer aplikacija oz. njen java *bytecode* teče v JVM za potrebe razhroščevanja,
- produkcijski način, kjer se aplikacija prevede v čisto JS in HTML kodo.

Omeniti velja še to, da obstaja že veliko odprtokodnih vtičnikov (*plugin*) za popularna razvijalska okolja (IDE) kot so *NetBeans, JDeveloper, InteliJ, Eclipse* itd.

3.2 Ogrodje Vaadin

Ogrodje *Vaadin* je odprtokodno ogrodje za razvijanje RIA. V nasprotju z ostalimi vrstami rešitev, kot so JS knjižnice (npr. *AngularJS*) ali različni vtiči (npr. *Flex*), je arhitektura *Vaadin*-a strežniško orientirana (*server-side*), kar pomeni, da glavna aplikacijska logika teče na strežniški strani. Tehnologija AJAX na odjemalski strani skrbi za interaktivno in bogato uporabniško izkušnjo (UX). Pravzaprav ogrodje *Vaadin* na svoji odjemalski strani koristi GWT ogrodje s katerim ga lahko tudi po svoje razširimo (lahko izdelamo svoje lastne komponente) [7].

Vaadin uporablja Java programski jezik za razvoj spletne vsebine. Uporablja dogodkovni programski model in vizualne gradnike (*widgets*), kar naredi programiranje bolj podobno programiranju GUI v drugih ogrodjih npr. *Swing*, *SWT*, kot pa »tradicionalnemu« razvoju spletnih aplikacij, kjer se prepleta veliko različnih tehnologij [7].

GWT se v ogrodju *Vaadin* rabi za prikaz (*front-side*), na strežniški strani pa *Vaadin* doda svojo preverjanje (*data validation*), kar ogrodje naredi bolj varno. Vgrajene komponente lahko vizualno spreminjamo z uporabo CSS-ja.

Aplikacije razvite s tem ogrodjem lahko tečejo na *servlet* vsebnikih kot dinamična spletna aplikacija ali pa kot *portleti*.

Torej vse kar potrebujemo za razvoj spletne aplikacije v *Vaadin*-u je programski jezik *Java, Vaadin*-ovo jar knjižnico(-ce) in že lahko razvijamo (RIA) aplikacije.

Glavne značilnosti:

- hiter razvoj,
- ploska krivulja učenja (sploh za razvijalce, ki so se že srečali s programskim jezikom *Java*),
- razvoj spletnih aplikacij podoben razvoju namiznih aplikacij,
- abstrakcija od specifičnih spletnih tehnologij,
- bogat nabor vgrajenih komponent,
- sestavljanje svojih komponent brez potrebe prevajanja (*Composite*)
- dobro zastavljen podatkovni model,

- dobra dokumentacija,
- podpira vse popularne IDE,
- teče na vseh sodobnih spletnih brskalnikih,
- videz uporabniškega vmesnika ločen od poslovne logike z uporabo tem (*theme*), ki se jih da razširiti.

3.3 Primerjava

Primerjavo sem naredil na nekaj kriterijih, ki so se mi, kot razvijalcu zdeli pomembni in so zbrani v naslednji tabeli (3-1):

	VAADIN	GWT
SPLOŠNO		
licenca	Apache 2.0	Apache 2.0
ogrodje lahko razširjamo	Da	Da
ogrodje lahko uporabljamo za komercijalne namene	Da	Da
namenjen za gradnjo SPA	Da	Da
število izdanih vzdrževalnih verzij v zadnjem letu	25	3
NAMEN		
namenjen za gradnjo dinamičnih strani	Ne	Ne
strežniško orientirana arhitektura	Da	Ne
odjemalsko orientirana arhitektura	Ne	Da

	VAADIN	GWT
končna aplikacija teče na	Java Servlet Vsebnik ali Java Portlet	Spletni strežnik
OKOLJE		
server-side API	Da	Ne
client-side API	Da	Da
programski jezik	Java in ostali JVM jeziki (npr. Skala)	Java
v celoti podpira tudi java 8	Da	Ne
LASTNOSTI		
dokumentacija	dobra	dobra
nivo abstrakcije od ostalih spletnih tehnlogij	10 / 10	6 / 10
avtomatizirana komunikacija server-brskalnik	Da	Ne
brskalnik ne rabi vtičnika	Da	Da
vgrajena Push (Websocket) podpora	Da	Ne
vgrajena WAI-ARIA podpora	Da	Da
ohranjanje stanja na strežniku (varnost)	Da	Ne
upravljanje z zgodovino brskalnika	Da	Da
i18n podpora	Da	Da
podpora lokalizaciji	Da	Da
TESTIRANJE		

	VAADIN	GWT
Avtomatizirano orodje za prevzemno testiranje	Da (TestBench)	Da
nudi orodje za testiranje UI (pixel level)	Da	Ne
Load testing s standartnimi orodji (JMeter, Gatling)	Da	Da
možnost testiranja UI koda z JUnit	Da	Da
LOOK 'N' FEEL		
vgrajen in parametrizirana podpora za teme	Da	Ne
razširljivo z CSS	Da	Da
razširljivo z Sass (SCSS)	Da	Da
število vgrajenih tem	5	4
KOMPONENTE		
Data grid komponenta	Da	Da
Spreadsheet komponenta	Da	Ne
Tree komponenta	Da	Da
Treetable komponenta	Da	Da
Charting komponenta	Da	Da
Drag n drop layouts and components	Da	Da
komponent optimizirane za mobilne naprave	Da	Da
podpora web komponentam	Da	Da

	VAADIN	GWT
število že vgrajenih (out of the box) komponent	10 / 10	5 / 10
SKUPONST		
skupnost in podpora	10 / 10	10 / 10
IDEs		
Eclipse plugin	Da	Da
IntelliJ plugin	Da	Da
Netbeans plugin	Da	Da
vizualni urejevalnik (povleci in spusti)	da (vendar beta različica; v novejši različici 7.6 že profesioanlni drag'n drop designer vendar je plačljiv)	-
INTEGRACIJA		
podpora Java EE integraciji	Da	Ne
podpora Spring integracijski	Da	Ne
OSGi kompatibilnost	Da	Ne
Maven upravljanje odvisnosti	Da	Da
Maven arhetipi	Da	Da
Ivy upravljanje odvisnosti	da	Ne
PODPRTI BRSKALNIKI		
IE6	Da	Ne
IE9+	Da	Da
Microsoft Edge	Da	Da

	VAADIN	GWT
Chrome	Da	Da
Firefox	Da	Da
Safari	Da	Da
Opera	Da	Da
Mobile Safari	Da	Da
Mobile Chrome	Da	Da
Mobile Windows Phone	Da	Da
RAZŠIRLJIVOST		
možnost razširjanja obstoječih komponent z Javo	kot kompozitne (Server-side) komponente (brez GWT prevajanja), ali pa kot komponente na server-side in client-side (GWT).	Da
možnost razširjanja obstoječih komponent z JavaScriptom	klici JS iz strežnika in kreacija JS komponent na odjemalski strani	Da
Nove razširitve pripravljene za uporabo	da, več kot 600 add-onov	da, kot knjižnice tretjih strank
ponudnik nudi podporo razširitvam tretjih strank	Da	Ne

Tabela 3-1:

Pri iskanju odgovora sem si pomagal s spletnimi viri, ki med drugim nudijo tudi primerjavo teh dveh ogrodij. Vsak navaja svoje kriterije za ocenjevanje in s tem seveda drugačne rezultate. To pa je odvisno predvsem od tega, kaj kdo išče pri določenem ogrodju.

3.4 Odločitev

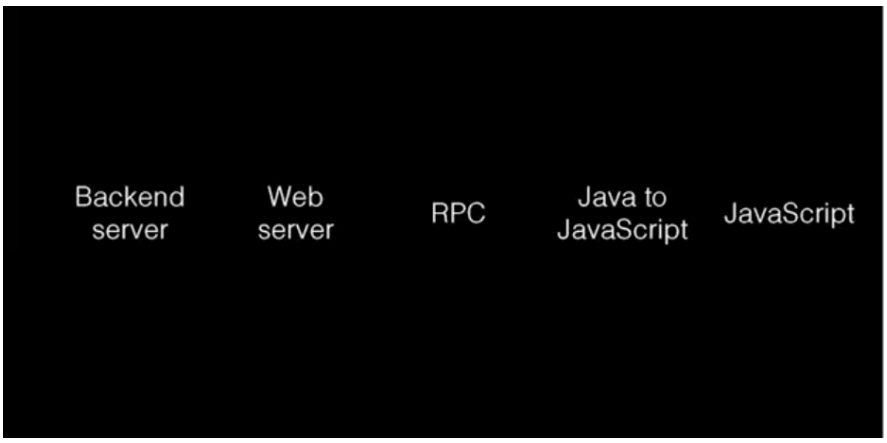
Z ostalimi razvijalci, ki so sodelovali na projektu smo se odločili za ogrodje *Vaadin* predvsem iz naslednjih razlogov.

Vaadin ima razvito veliko število visoko kakovostnih in kompleksnih komponent. Torej če želimo takoj začeti programirati poslovno aplikacijo je tu v veliki prednosti *Vaadin*, ker GWT nima veliko razvitih komponent. V nasprotnem primeru bi morali uporabiti kakšno drugo »plačljivo« knjižnico (npr. *Sencha GXT*), ki razširja GWT in vsebuje veliko več komponent kot sam GWT. *Vaadin* pa lahko brezplačno uporabimo tudi v komercialne namene [4].

Programska izkušnja pri uporabi ogrodja *Vaadin* je podobna, kot če bi programirali namizno aplikacijo.

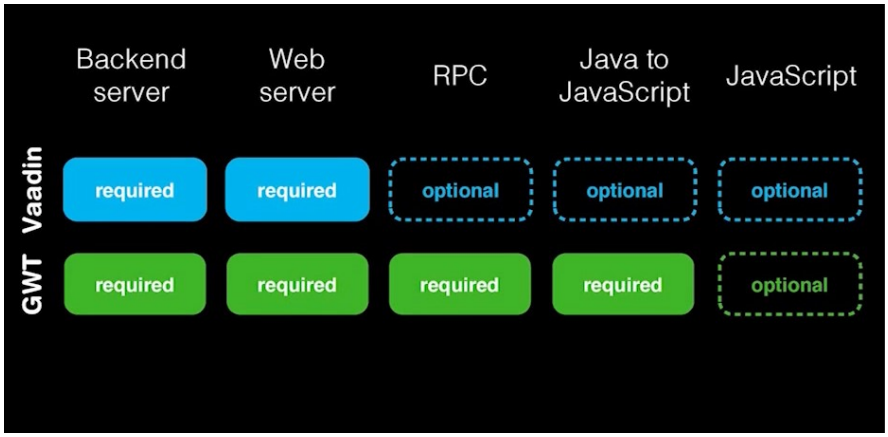
Razdelimo poslovno RI aplikacijo na posamezne plasti abstrakcije iz razvijalskega stališča tj. »Kaj vse moramo implementirati za RIA?«. Tipično dobimo pet plasti, iz katerih je aplikacija sestavljena (Slika 3-1: plasti abstrakcije RIA s stališča razvijalca.Slika 3-1):

- poslovni nivo in baza (oz. *Backend server*),
- spletni nivo oz. vstopna točka za našo UI kodo (*Web server*),
- komunikacija med strežnikom in odjemalcem (RPC),
- java koda prevedena v JS (*Java2JS*)
- čista JS koda (JS).



Slika 3-1: plasti abstrakcije RIA s stališča razvijalca.

V primeru uporabe GWT ogrodja bi morali implementirati 4 plasti: *Backend server, Web server, RPC* in *Java2JS*. Če pa uporabimo *Vaadin* ogrodje pa moramo implementirati samo 2 plasti: *Backend server in Web server*, za ostalo poskrbi *Vaadin* sam (Slika 3-2).



Slika 3-2: implementacija posameznih plasti abstrakcije RIA glede na ogrodje.

Če na kratko povzamemo prednosti, ki so nas prepričale v uporabo *Vaadin* ogrodja. Ogrodje vsebuje strežniški razvojni model, ki:

- zmanjša število vrstic kode (z zmanjšanjem plasti, ki jih moramo implementirati za uporabniški vmesnik),
- omogoča uporabo katerega koli jezika, ki bazira na JVM (v primeru razširitve razvojnega tima),
- poslovna logika kompletno teče na strežniški strani (poveča varnost),
- dopušča sinhrone klice v katerikoli strežniški API,
- lahko uporabimo katerokoli *Java* standardno knjižnico in orodja strani uporabniškega vmesnika,
- ne zahteva koraka *Java* v *JavaScript* prevajanja (ki, je obvezen za GWT pred izdajo (*deploy*) aplikacije in je časovno zelo zahtevna operacija),
- ima že vgrajeno *server push* podporo (tj. ko strežik lahko osveži katerikoli klient brez zahteve klienta oz. dogodka, ki ga je povzročil klient),

- podpora temam (vgrajene teme *Reindeer, Valo, Chameleon in Runo*),
- možnost razširjanja tem po meri (demo.vaadin.com/ReindeerTheme, vaadin.com/valo, demo.vaadin.com/valo-theme, demo.vaadin.com/chameleon-editor).

Na koncu omenim še to, da v trenutku razvijanja aplikacije nismo imeli veliko prostih človeških virov, časa za razvoj pa zelo malo. Na voljo je bilo nekaj DB administratorjev in razvijalcev na *Oracle* bazi, ki so poznali ISCSD tematiko in nekaj prostih razvijalcev, ki so povečini programirali na MS platformi (predvsem v *C#,* nekateri uporabljali tudi *Javo*). Z izbiro *Vaadin* ogrodja smo hitro vse vpeljali v razvoj modula, tudi če prej niso imeli veliko znanja o Java programskem jeziku (jim je pa prišlo prav znanje razvijanja npr. v *event driven* modelu).

4. Ogrodje Vaadin

V tem poglavju bomo na kratko opisali ogrodje *Vaadin*, njegov namen, arhitekturo ogrodja in njegove lastnosti, ter kaj nam ogrodje ponuja kot razvijalcem aplikacij. Razlikujemo pravzaprav dva modela programiranja aplikacij v *Vaadin* ogrodju: programiranje aplikacije na strežniški strani (*server side Application*) in programiranje aplikacije na strani odjemalca (*client side Application*) aplikacij. model programiranje na strani odjemalca pravzaprav omogoča podobno programiranje kot GWT, z določenimi razširitvami *Vaadin* ogrodja. Če v nadaljevanju posebej ne omenimo kateri model imamo v mislih, govorimo o modelu programiranja aplikaciji na strežniški strani.

4.1 Osnovni namen ogrodja

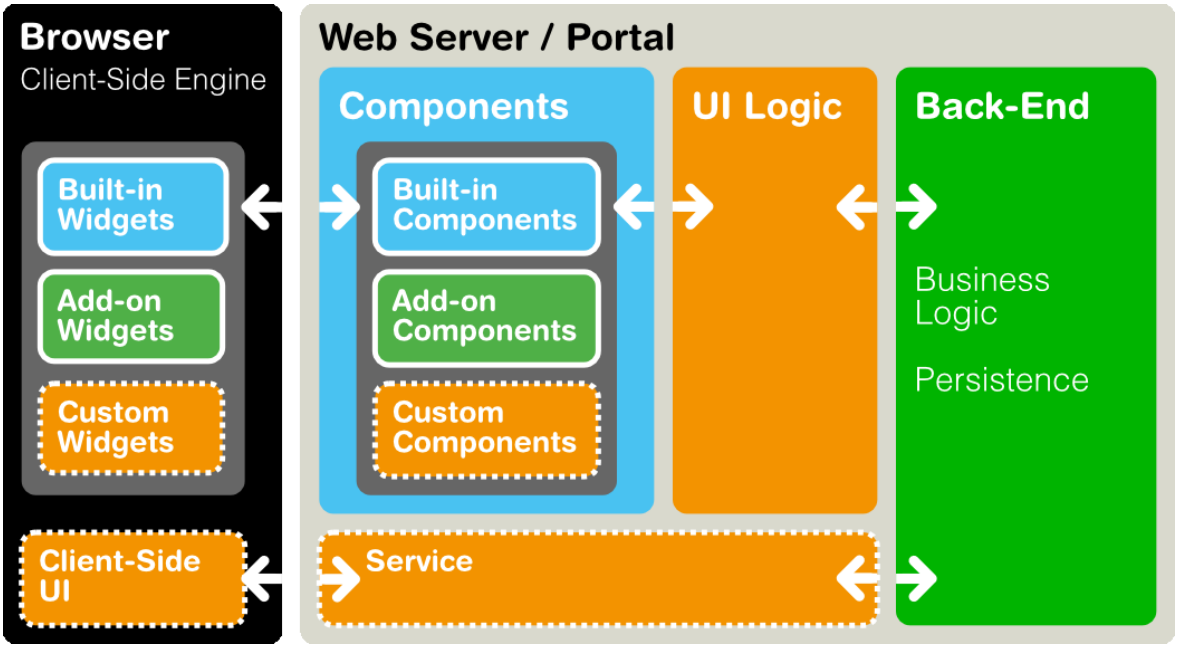
Vaadin je namenjen hitremu razvoj bogatih internetnih aplikacij z *Java* programskim jezikom in omogoča visoko produktivnost in kakovost pri razvoju majhnih in srednje velikih projektov. Torej lahko rečemo, da je glavno poslanstvo ogrodja ustvarjanje bogatih spletnih uporabniških vmesnikov za poslovne aplikacije. Nikakor ni ogrodje namenjeno za izdelovanje spletnih strani, za kar obstaja veliko drugih, boljših ogrodij.

Ogrodje je sestavljeno iz API na strani strežnika, API na strani klienta/odjemalca in velike palete UI komponent in vizualnih gradnikov. Z njimi nadzorujemo videz komponent in podatkovnega modela, ki omogoča direktno povezavo komponente na strežniški strani na podatkovni vir. Za razvoj lastnih komponent na strani klienta pa ima Vaadin tudi prevajalnik, ki omogoča prevajanje *Jave* v *JavaScript*.

Na strežniški strani aplikacija teče kot *servelt* v *servlet* vsebniku (*servlet Container*) in streže HTTP zahtevam, tako da prejme zahtevo in jo interpretira kot dogodek za točno določeno uporabniško sejo. Dogodki so v povezavi z določeno UI komponento (na strežniški strani) in se predajo naprej poslušalcem dogodkov, definiranih v aplikaciji in v končni fazi upravljavcu dogodkov (*event-handler*). Če pride do spremembe pri komponentah na strežniški strani, *servlet* generira odgovor za klient stran in ga posreduje odjemalcu (točneje *server-side* mehanizmu *com.vaadin.client.ApplicationConnection*), ta pa na podlagi zahteve (*request*) ali dogodka

(*event*) naredi spremembe prikaza strani v brskalniku. To omogoča takojšnje osveževanje UI na klient-strani neposredno iz drugih niti aplikacije [8].

Na naslednji sliki (Slika 4-1) je grafično prikazana osnovna arhitektura spletnih aplikacij, ki so narejene s pomočjo ogrodja *Vaadin*.



Slika 4-1: osnovna arhitektura *Vaadni* spletnih aplicacij.

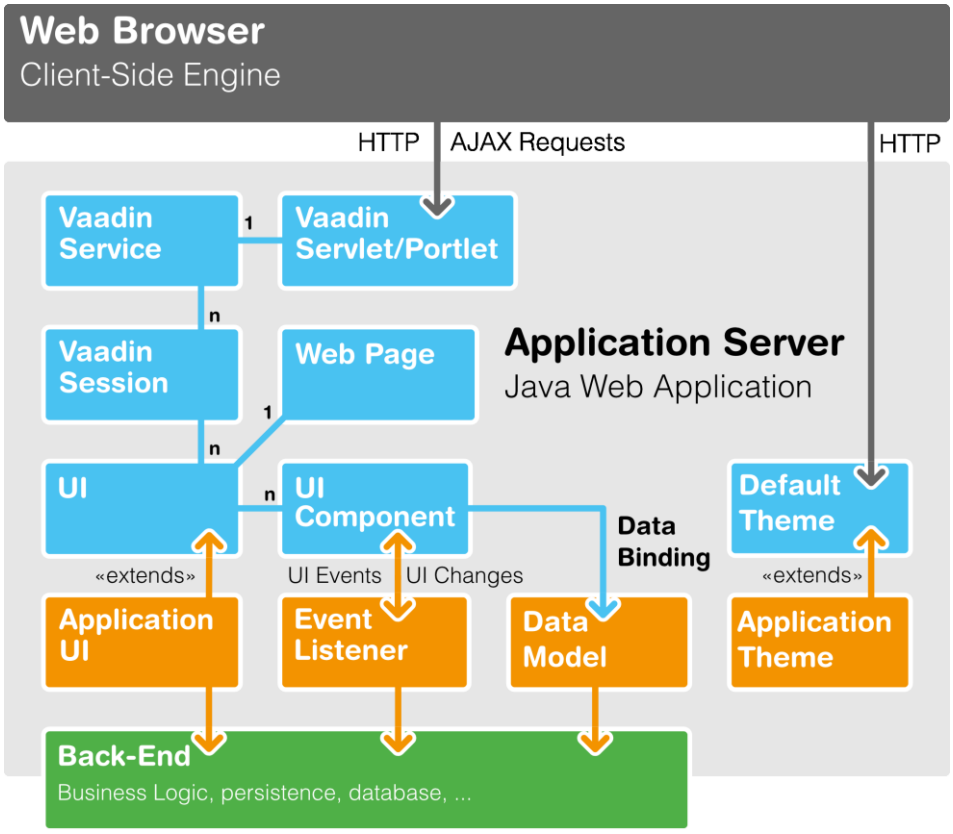
V nadaljevanju so opisani glavni deli arhitekture ogrodja *Vaadin*.

4.2 Uporabniški vmesnik (UI)

Uporabniški vmesnik (UI) omogoča interakcijo uporabnikov z poslovno logiko in podatki aplikacije. UI je izveden kot UI razred, ki razširja razred *com.vaadin.ui.UI*, to je tudi glavna vstopna točka za razvijalce, kjer dodajajo svojo kodo (Slika 4-2). Glavna naloga razreda je, da zgradi vmesnik iz UI komponent in nastavi poslušalce dogodkov, ki obravnavajo uporabnikovo interakcijo z vmesnikom in se kot tak prenese v brskalnik.

```
public class HelloWorld extends UI {
    protected void init(VaadinRequest request) {
        //... inicializacijska in vstopna točka obenem za našo kodo ...
    }
}
```

Slika 4-2: Inicializacija aplikacije in vstopna točka za našo kodo.



Slika 4-3: Arhitektura aplikacije na strežniški strani.

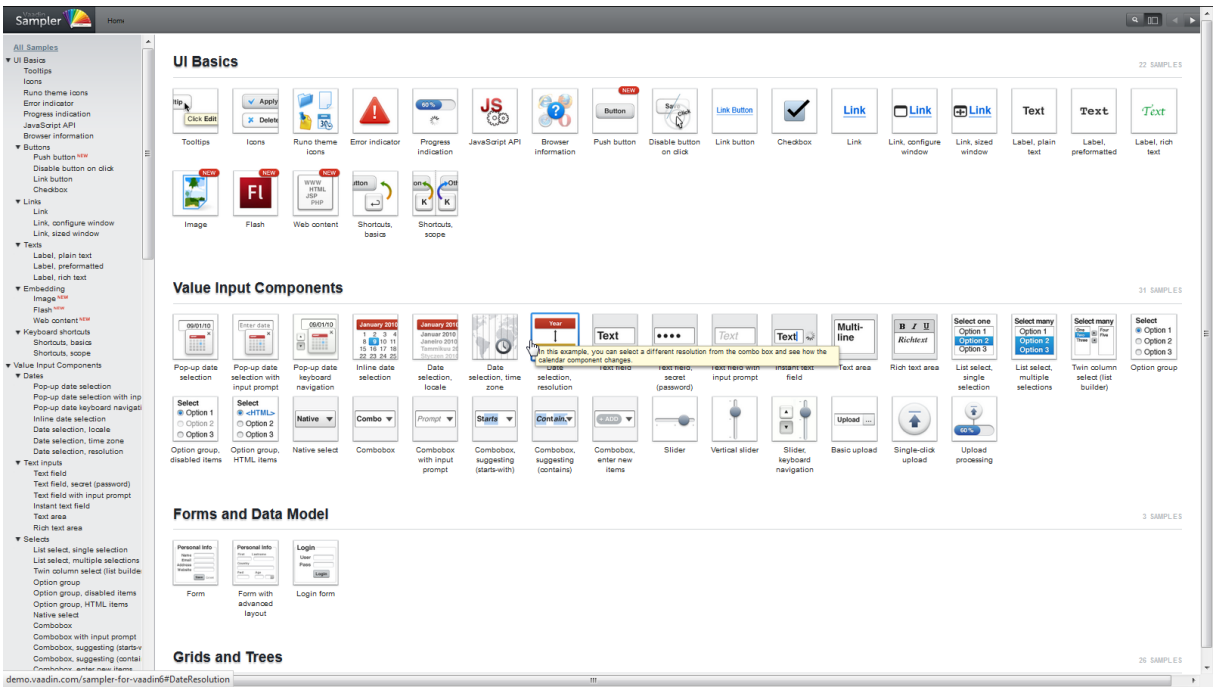
Temeljni elemnti aplikacije poleg UI (Slika 4-3):

- *UI*: objekt, ki prestavlja HTML *fragment* naše aplikacijo oz. celotno spletno stran *Vaadin* aplikacije.
- *Page*: objekt, ki predstavlja s spletno stran in celotno okno spletnega brskalnika, kjer teče UI.
- *VaadinSession*: objekt, ki predstavlja uporabniško sejo z enim ali več odprtimi UI.
- *UI komponente*: komponente vmesnika in omogočajo interakcijo z uporabnikom.

- *Dogodki in poslušalci dogodkov*: ti se odzivajo na uporabnikove interakcije z UI komponentami.
- *Viri*: slike (prikaz), povezave in dokumenti (*download*), ki sestavljajo našo aplikacijo.
- *Teme*: skrbijo za izgled UI komponent in aplikacije.
- *Povezovanje s podatki*: povezava komponent na podatkovne vire / prikaz podatkov.

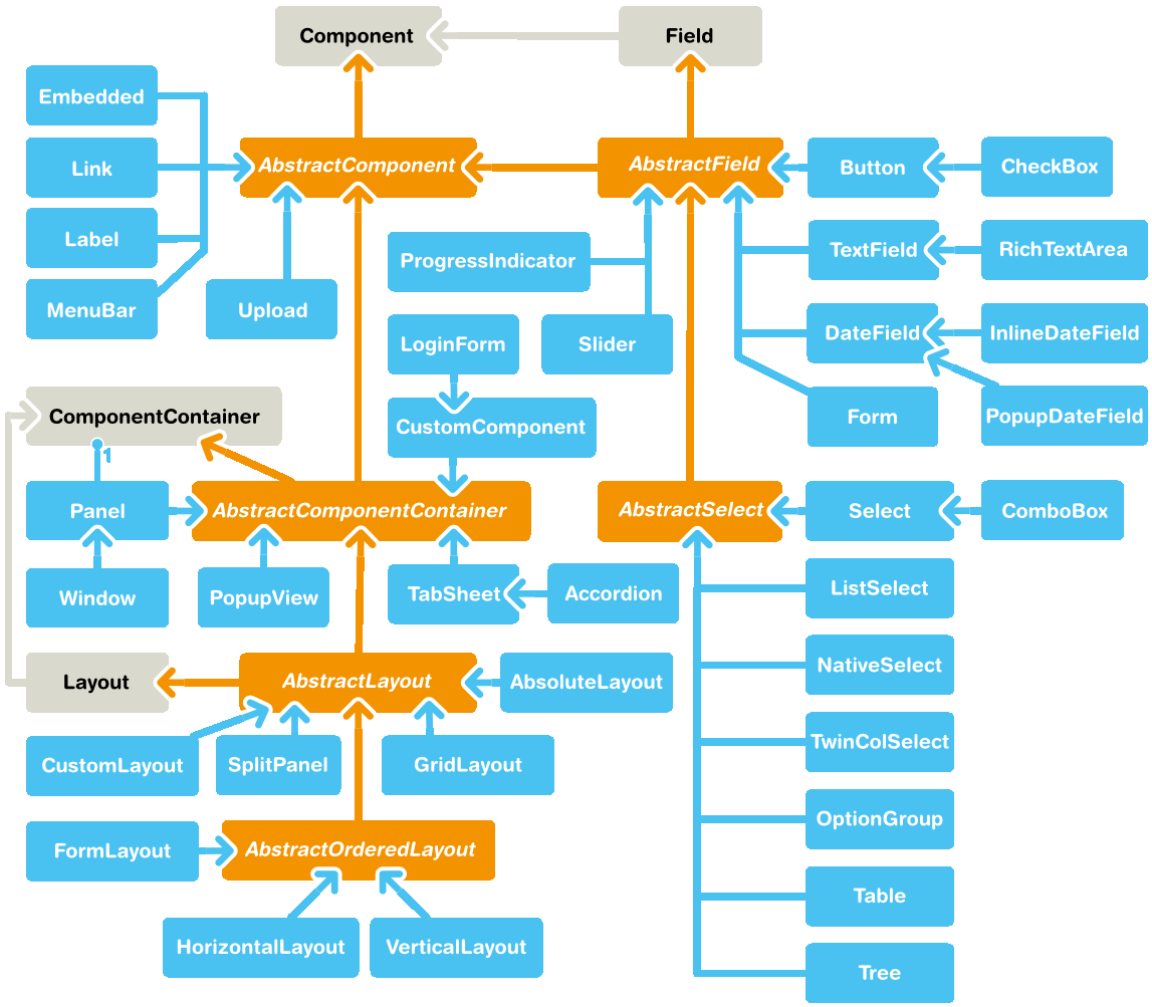
4.3 Komponente UI

Uporabniški vmesnik aplikacije je sestavljen iz komponent, ki jih kreira in razporedi aplikacija. Vsaka komponenta na strežniški strani ima neke vrste dvojnika na strani klienta, tj. vizualni gradnik (*widget*) prikazan v brskalniku, s katerim komunicira uporabnik (prek dogodkov- klik gumba, sprememba vrednosti polja itd.). Vizualni gradniki na strani klienta pa lahko komunicirajo samo s aplikacijo na strani klienta (kot GWT *widgets*). Komponente na strežniški strani dogodke posreduje aplikacijski logiki.



Slika 4-4: Vgrajene UI komponente.

Na naslednji sliki (Slika 4-5) je prikazana hierarhična ureditev strežniških UI komponent v ogrodju *Vaadin*.

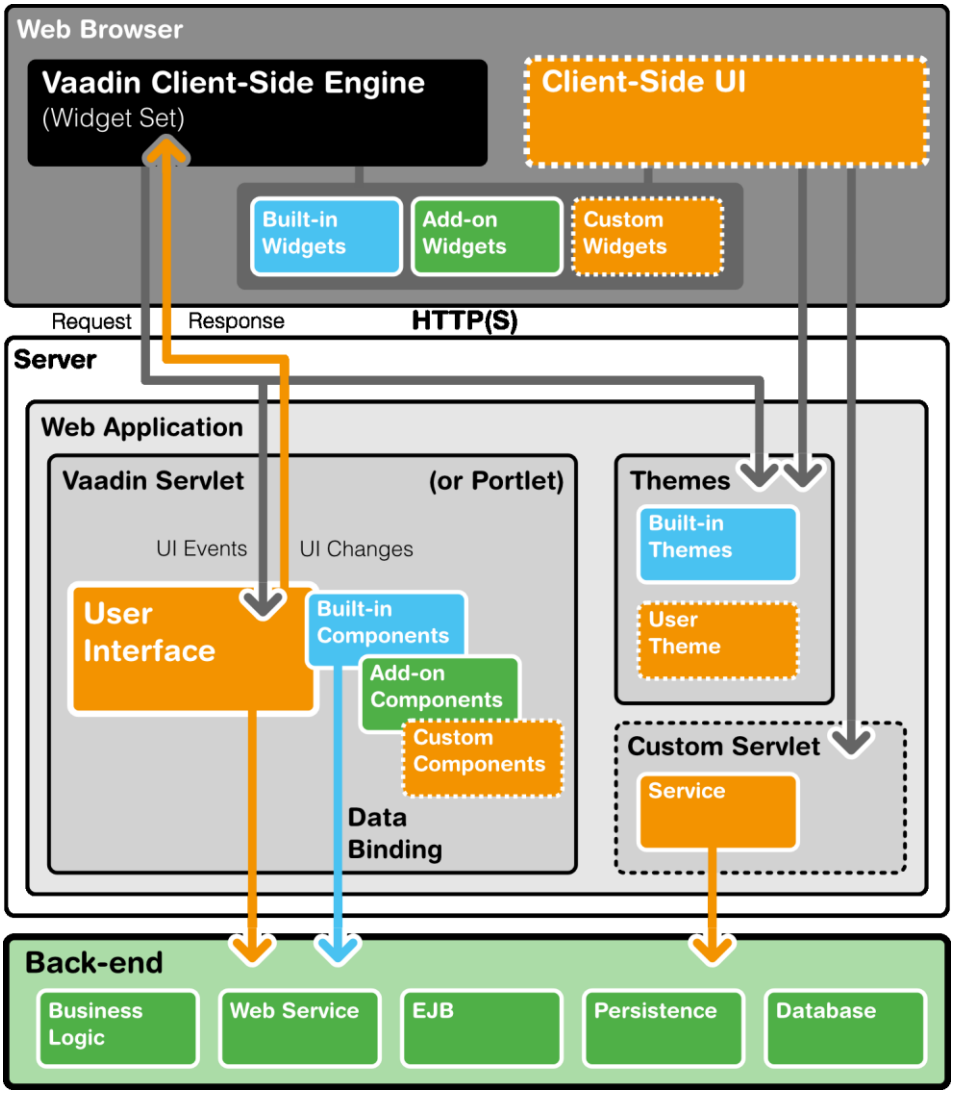


Slika 4-5: Razredna hierarhija UI komponent

4.4 Mehanizem strani klienta (Client-Side Engine)

Mehanizem na strani klienta upravlja z upodabljanjem UI v odjemalcu, tj. spletnem brskalniku z uporabo številnih vizualnih gradnikov na strani klienta, dvojnikov strežniških komponent. Uporabnikovo spremembo/transakcijo sporoči strežniški strani in po prejetju odgovora, ustrezno posodobi vmesnik. Komunikacija poteka preko asinhronih HTTP ali HTTPS zahtev.

Na naslednji sliki (Slika 4-6) je grafično prikazana strežniška stran, klient-stran in komunikacija med njima po tem ko je stran že naložena v brskalniku (tj. izvajalno okolje) [8].



Slika 4-6: Ilustracija *Vaadin* izvajalnega okolja.

4.5 Razred VaadinServlet

Strežniške *Vaadin* aplikacije temeljijo na *Java Servlet API*-ju. *Servlet*[1] (oz. razred *VaadinServlet*), sprejema zahtevke od različnih klientov, ugotovi kateri uporabniški seji pripada (s pomočjo piškotov) in prenese zahtevo ustrezni seji.

[1] Poseben Java razred (po specifikaciji JSR-000340 Java Servlet 3.1 Final Release), s katerim razširimo funkcionalnosti strežniških programov, ki delujejo po načelu zahteva-odgovor.

4.6 Teme

Vaadin omogoča ločevanje med videzom in strukturo komponent UI. Ogrodje UI logiko obravnava kot *Java* kodo, prikaz vmesnika pa je definiran s *CSS* ali *Sass* temami. Uporabniške teme lahko poleg slogov, vključujejo HTML predloge, ki določajo postavitev po meri in druge vire, kot so slike in oblika pisave.

4.7 Dogodki

Interakcija z UI (na klient strani) ustvarja dogodke, ki jih najprej obdelajo vizualni gradniki na strani odjemalca, nato pa se prenesejo na strežnik, v *VaadinServlet* in od tam do UI komponent na strežniški strani, in naprej do poslušalcev dogodkov definiranih v aplikaciji.

4.8 Potiskanje podatkov v smeri strežnik klient (Server Push)

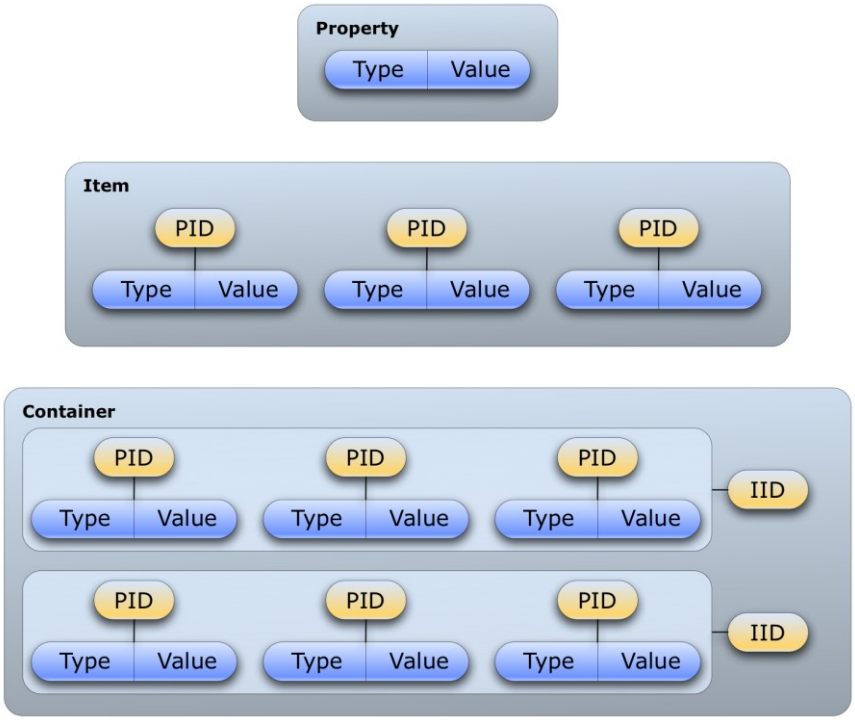
Poleg dogodkovnega programskega modela, *Vaadin* podpira tudi ti. *server push* način, kjer se spremembe na UI prenesejo iz strežnika na klient brez predhodne zahteve s strani klienta. Tako se lahko implementira takojšnje osveževanje iz drugih niti in UI, brez predhodnega čakanja na zahtevo (npr. indeks delnic, ali pa novo sporočilo (socialna omrežja)).

4.9 Povezovanje s podatki (Data Binding)

Ogrodje zagotavlja tudi močan podatkovni model za povezovanje podatkov predstavljenih v UI komponentah tipa *com.vaadin.ui.AbstractField* (npr. tekstovno polje, ali pa izbirno polje), s podatkovnimi viri. Tako lahko UI komponente spreminjajo in posodabljajo podatke brez kakršne koli dodatne kode. Vse komponente ki razširjajo ta tip interno uporabljajo ta podatkovni model. Npr. tabela se lahko veže na podatkovni model *SQL* poizvedbe ali pa na podatkovni model xml datoteke.

Na sliki (Slika 4-7) je prikazana gnezdena tri nivojska hierarhija podatkovnega modela. *Property*, *Item* in *Container*. Če potegnemo analogijo na primer s preglednico je bi celica ustrezala objektu *Property*, vrstica (sestavljena iz celic) bi predstavljala objekt *Item* in celotna tabela (sestavljena iz vrstic) bi predstavljala objekt *Container*.

Vaadin-ov podatkovni model je eden od glavnih konceptov ogrodja.



Slika 4-7: podatkovni model ogrodja Vaadin.

4.10 Aplikacije na strani klienta (Client-Side Applications)

Kot smo že omenili ogrodje omogoča tudi razvoj modulov, ki tečejo na strani klienta v brskalniku. Moduli na strani klienta lahko uporabljajo enake vizualne gradnike, teme in storitve v ozadju kot aplikacije na strani strežnika. Koristni so, ko rabimo zelo odziven uporabniški vmesnik npr. pri igrah ali pa pri serviranju velikega števila klientov s strežniško kodo, ki ne hrani stanja ali pa v primeru nudenja nepovezanega (*off-line*) načina strežniški aplikaciji.

4.11 Zaledni del (Back-end)

Namen ogrodja je gradnja UI, zato je priporočljivo, da so ostali aplikacijski nivoji ločeni od UI logike. Poslovno logiko tako ločimo do UI kode z uporabo različnih API-jev npr. EJB-ji, ali pa oddaljene storitve, ki tečejo v ozadju. Shranjevanje podatkov pa implementiramo po navadi z uporabo DBMS, do katerih dostopamo prek JPA.

5. Implementacija rešitve

Končna rešitev oz. »modul za poročanje« (Slika 5-2) je razvit kot spletna aplikacija (RIA) na tri nivojski arhitekturi, implementacija uporabniškega vmesnika pa je narejena s pomočjo ogrodja *Vaadin ver. 7*.

Za podatkovno zbirko je uporabljena obstoječo ISCSD zbirka, ki jo poganja strežnik *Oracle Database 10g*.

Kot aplikacijski strežnik smo uporabili obstoječi *JBoss AS ver. 6.0.0 Final "NEO"*, na katerem že tečejo aplikacije: zunanji del MDDSZ-Portal-a, VPPJS, itd. Modul za poročanje je dostopen zunaj HKOM omrežja, ker vsi subjekti, ki morajo poročati o dodeljenih štipendijah, nimajo dostopa do HKOM-a. Zagotovljena je tudi povezava aplikacije s podatkovnim strežnikom in zbirko, ker na aplikacijskem strežniku že tečejo aplikacije, ki uporabljajo navedeno podatkovno zbirko.

Aplikacija se izvaja v brskalniku. Podprti so vsi novejši brskalniki: *Internet Explorer, Google Chrome, Mozilla Firefox, Opera, Safari*.

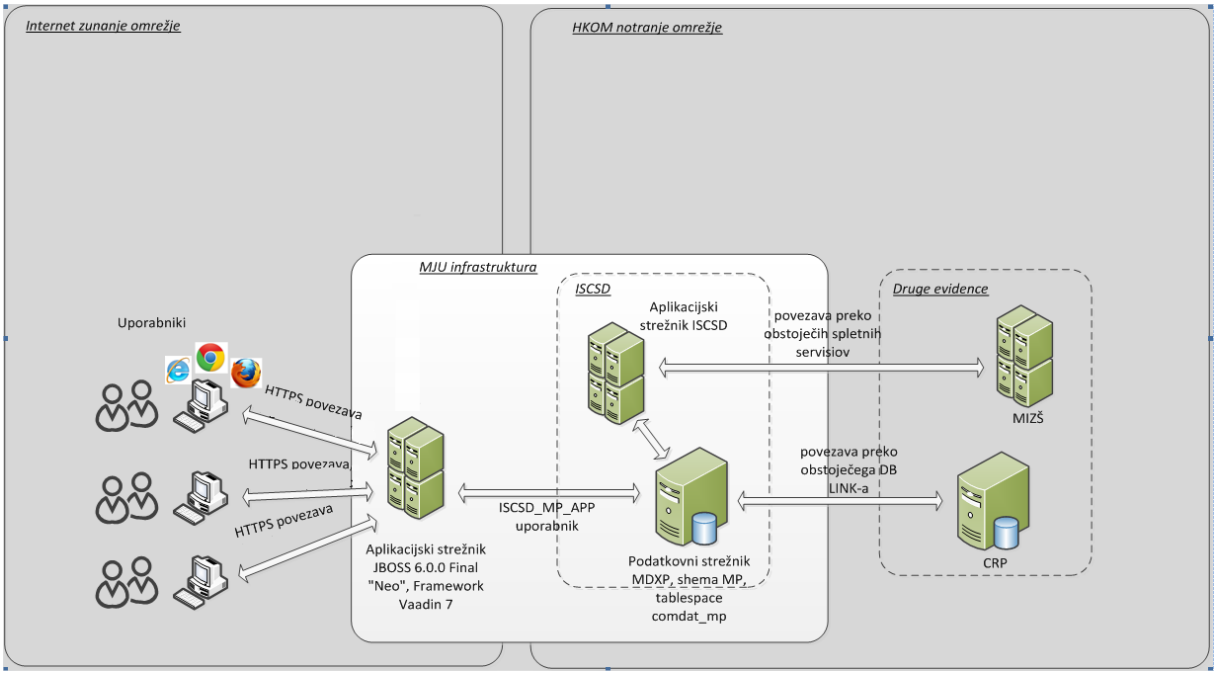
Narejena je tudi nova tema definirana iscsd_mp.scss s katero se upravlja izgled aplikacije, ki je razširitev teme vgrajene teme *Valo*.

Da se zadosti zahtevam ZŠtip-1 in zahtevam SURS-a nov modul, z ustreznimi povezavami na različne notranje in zunanje vire, omogoča pregled nad podatki in:

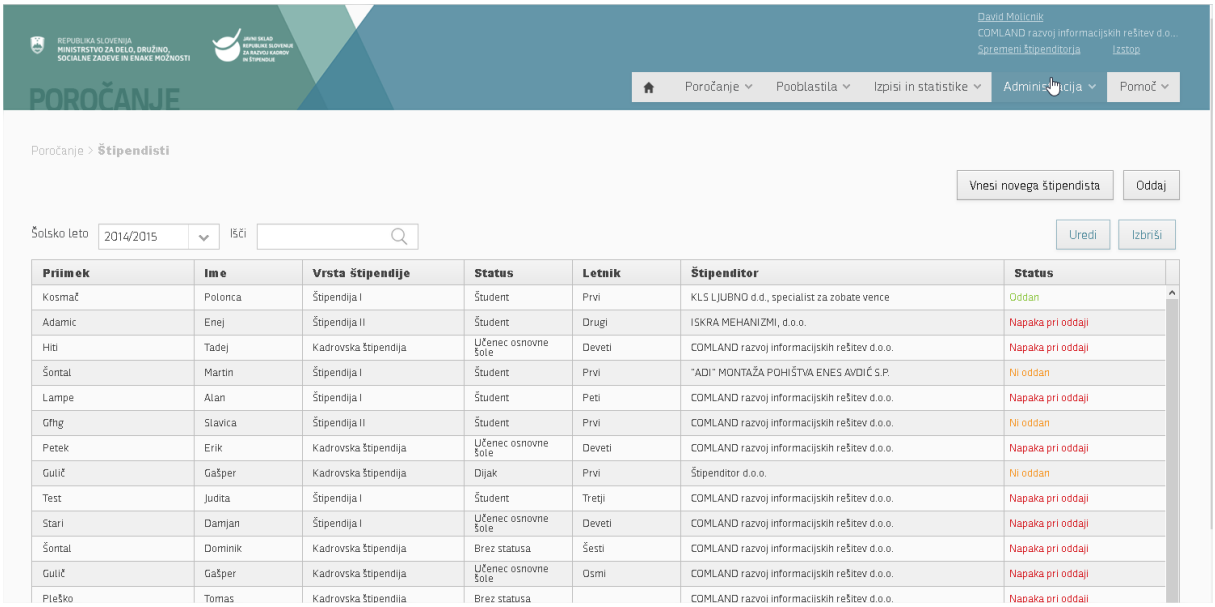
- poročanje delodajalcev Skladu o potrebah po kadrovskih štipendijah v naslednjem šolskem/študijskem letu,
- poročanje o dodeljenih štipendijah, ki jih dodeljujejo subjekti, ki dodeljujejo štipendije po drugih predpisih (4. člen ZŠtip-1) Skladu: sofinancirane kadrovske (posredne), kadrovske (tiste, ki jih delodajalci izplačajo sami v celoti, torej vsi s.p., d.o.o. …),
- pregledovanje nad podatki in poročanje o štipendijah, ki jih dodeljuje Sklad,

- poročanje SURS-u s strani Ministrstva o vseh dodeljenih štipendijah v preteklem letu.

Shema produkcijskega okolja aplikacije je podana na naslednji sliki (Slika 5-1) .



Slika 5-1:Shema okolja, v katerem teče aplikacija.



Slika 5-2: Prikaz končne aplikacije v spletnem brskalniku.

5.1 Testno in produkcijsko okolje

Na strani naročnika sta bili vzpostavljeni 2 okolji: testno in produkcijsko (podobno kot za ISCSD sistem). Na obeh okoljih so bile nameščene enake verzije ogrodja, aplikacijskega strežnika in podatkovne baze.

Aplikacija je tako dostopna na naslovih:

- https://emddsz-test.gov.si/iscsd_mp
- https://emddsz.gov.si/iscsd_mp

Testiranje aplikacije se je najprej izvajalo na testnem okolju. Aplikacija na testnem okolju je dostopna samo za uporabnike, ki dostopajo znotraj HKOM omrežja. Testiranje produkcijske aplikacije se je izvajalo na produkcijskem okolju in ga je izvajal izvajalec in naročnik. V fazi testiranja produkcijske aplikacije je bil dostop zunanjim uporabnikom onemogočen.

5.2 Razvojno okolje

Na strani izvajalca smo postavili testno okolje z enako namestitvijo kot na naročniški strani. Vsak razvijalec na aplikacijskem nivoju pa je imel nameščeno svoje razvojno okolje in dostop do skupne podatkovne zbirke.

Sam sem za razvoj smo uporabljali naslednja orodja:

- *Eclipse Luna IDE*
- *Vaadin for Eclipse plugin*
- *Oracle Database 10g R2*
- *Toad for Oracle 12.7*
- *Balzamiq Mockups 2*
- *GIT*
- *Portacle*

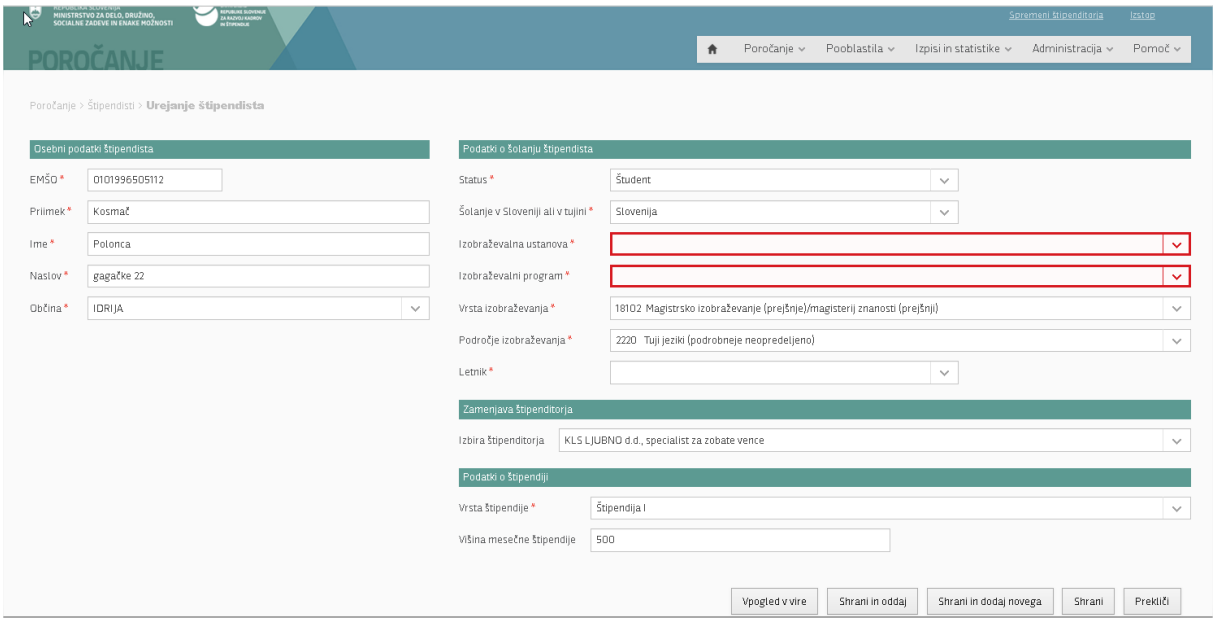
5.3 Predstavitev rešitve

Vsebinsko je aplikacija sestavljena iz naslednjih delov:

- Del za prijavo v aplikacijo
- Del za poročanje o štipendiranju
- Del za urejanje pooblastil
- Administrativni del

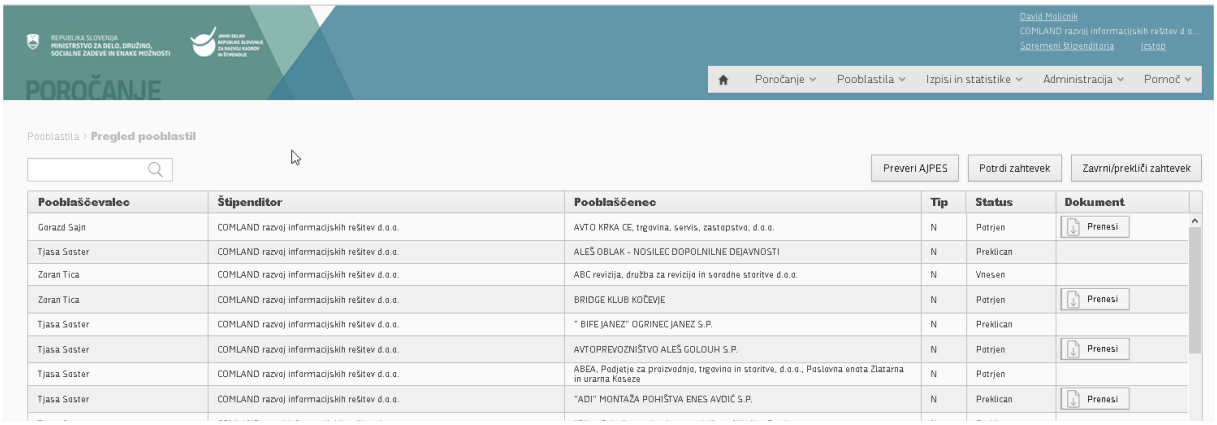
V delu za prijavo v aplikacijo se uporabnik prek varne povezave prijavi v aplikacijo. V temu delu je določen uporabnik aplikacije in organ, za katerega uporabnik ureja podatke.

V delu za poročanje o štipendiranju je omogočen vnos in spreminjanje podatkov o štipendiranju (štipendisti, zneski, obdobja štipendiranja...) (Slika 5-3) ter vpogled v te podatke in različni izpisi/poročila.



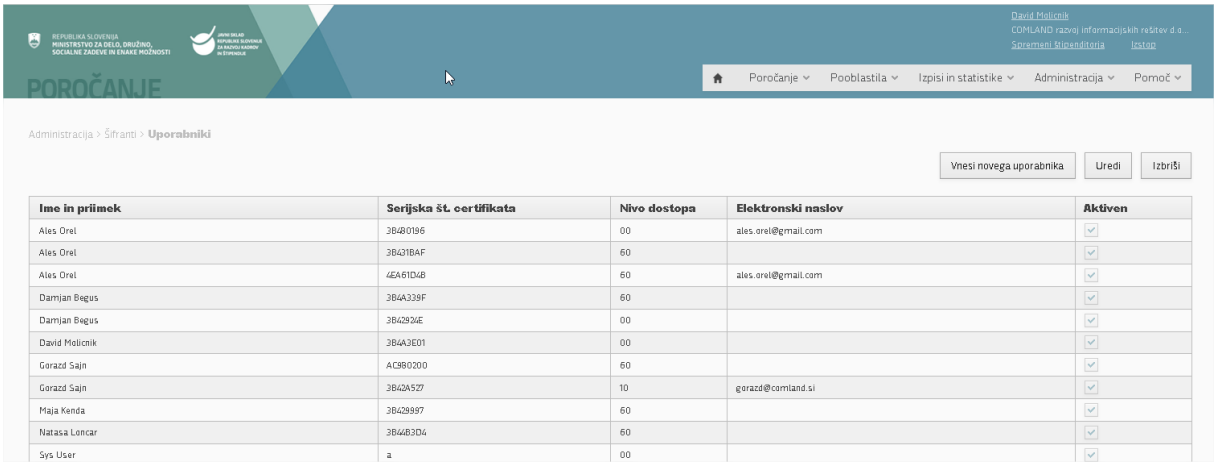
Slika 5-3: Maska za vnos štipendista.

V delu za urejanje pooblastil je omogočen vnos podatkov o pooblastilu (Slika 5-4), ki pooblašča eno pravno osebo/fizično osebo/organ, da ima pravice urejati poročanje o štipendiranju v imenu druge pravne osebe/organa.



Slika 5-4: Urejanje pooblastil.

V administrativnem delu je omogočeno urejanje in spreminjanje sistemskih parametrov, potrebnih za delovanje aplikacije ter urejanje šifrantov (Slika 5-5).



Slika 5-5:Maska administracijskega modula

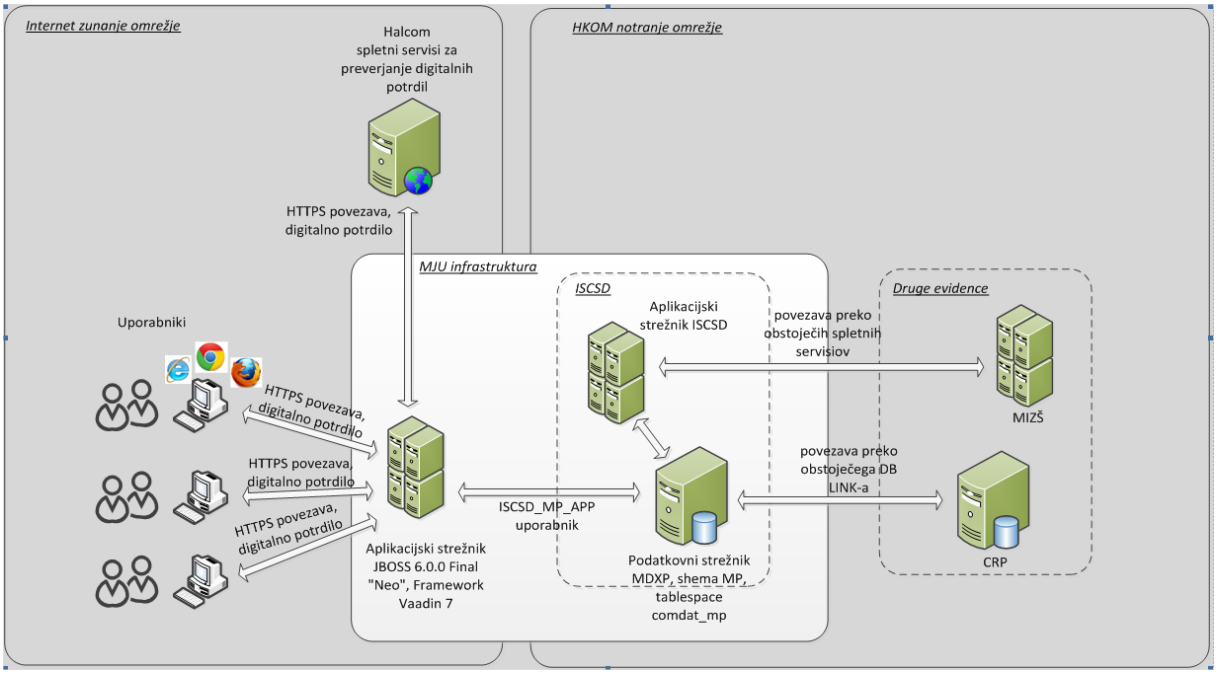
Ker je aplikacija dokaj obsežna, se bomo z namenom prikazati uporabo ogrodja v praksi v naslednjem poglavju omejili na del za prijavo v aplikacijo.

6. Praktična uporaba

Podatki, ki se zbirajo v modulu so zaupne narave. Hkrati mora biti aplikacija javno dostopna vsem, ki morajo poročati o štipendijah. Zato je bilo potrebno zagotoviti ustrezno stopnjo varnosti v aplikaciji, ki se začne z primerno identifikacijo in avtentikacijo.

Danes se za zagotavljanje avtentikacije uporabljajo trije tipi informacij: nekaj kar veš (PIN, geslo), nekaj kar imaš (vozniško dovoljenje, osebna izkaznica), ali nekaj kar si (biometrija – branje mrežnice ali prstnih vtisov) [9]. V našem primeru je dovolj močan tip avtentikacije prijavljanje uporabnikov z digitalnimi potrdili (nekaj, kar imaš).

Ker ogrodje *Vaadin* na strežniški strani omogoča vpeljavo katerekoli *Java* knjižnice in s tem razširitev funkcionalnosti naše aplikacije, se odločimo za povezavo modula s *Halcom* -ovimi spletnimi storitvami preverjanja certifikatov. Tako se shema okolja dopolni s preverjanjem certifikatov. (Slika 6-1).



Slika 6-1:Dopolnjena shema s preverjanjem certifikatov.

Prednost takšne izvedbe avtentikacije je poleg večje varnosti tudi to, da odpade vsakršno rokovanje (hranjenje in izdajanje) z uporabniškimi imeni in gesli. Vse kar rabi uporabnik za dostop do aplikacije je veljavno digitalno potrdilo v svojem brskalniku in internetni dostop.

V primeru, da uporabnik nima digitalnega potrdila lahko podatke še vedno odda s pomočjo obrazca, ki ga lahko dobi na spletni strani Sklada.

Ker obstaja veliko računovodskih podjetij, ki vodijo podatke o drugih podjetjih (subjektih), se pričakuje tudi oddajanje podatkov v imenu drugih subjektov na podlagi predhodnega pooblastila.

6.1 Organizacija uporabnikov

V modulu za poročanje so trije tipi uporabnikov:

- uporabnik podatkovne baze
- uporabnik za dostop aplikacije do podatkovne baze
- uporabniki aplikacije

Uporabnik podatkovne baze je poimenovan MP. Uporabnik je kreiran na nivoju podatkovne baze in ima ustrezno podatkovno shemo (poimenovana enako kot uporabnik - "MP"), kjer se nahajajo potrebni objekti v podatkovni bazi (tabele, sekvence, bazna programska logika...).

Uporabnik za dostop aplikacije do podatkovne baze je poimenovan ISCSD_MP_APP. Preko njega aplikacija dostopa do podatkovne baze in izvajala podatkovne operacije (branje, dodajanje, brisanje, spreminjanje, uporaba bazne programske logike). Dostop tega uporabnika do drugih objektov, ki so del ISCSD je zelo omejen, urejen prek *oraclovih* sistemskih dovoljenj (*system grant*).

Uporabniki aplikacije so fizični uporabniki, ki uporabljajo aplikacijo. Njihovi podatki so shranjeni v okviru MP sheme. Za dostop do podatkovne baze pa se uporablja ISCSD_MP_APP uporabnik.

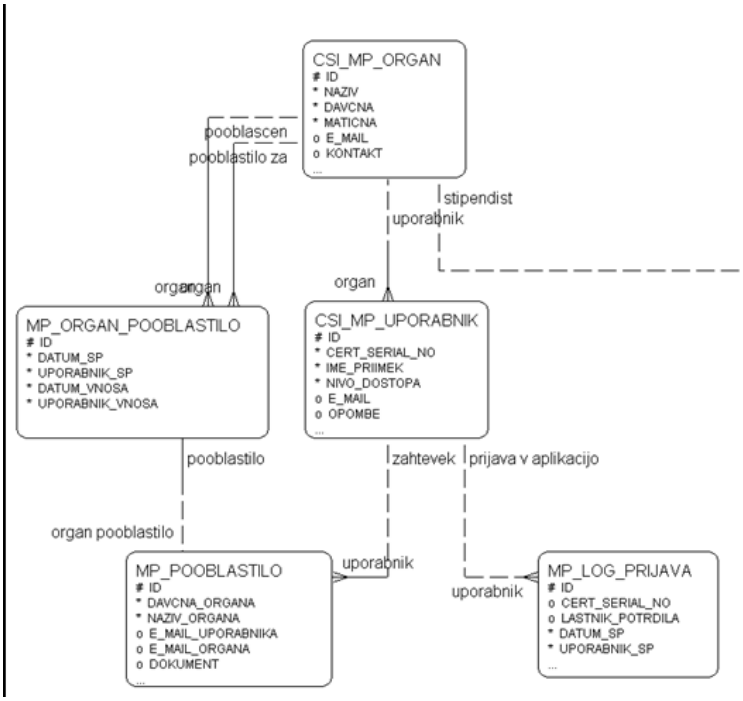
6.1.1 Podatki uporabnika in organa

Podatki uporabnika aplikacije so shranjeni v podatkovno bazo v poseben šifrant uporabnikov CSI_MP_UPORABNIKI (Slika 6-2). V šifrantu se nahajajo naslednji podatki:

- ime in priimek (določeno samodejno iz digitalnega potrdila)
- elektronski naslov
- opomba
- pravica dostopa (00, 10 60)
- organ, kateremu uporabnik pripada

Šifrant organov je CSI_MP_ORGANI in vsebuje naslednje podatke:

- naziv organa
- davčna številka
- matična številka
- kontaktni podatki (naslov, telefon, e-mail...)



Slika 6-2: ER diagram uporabnikov in organov.

6.1.2 Pravice dostopa uporabnikov

V aplikaciji obstajajo trije nivoji dostopa:

- nivo dostopa 60 - poročevalci
- nivo dostopa 10 - Sklad in Ministrstvo
- nivo dostopa 00 - administrator

Poročevalci imajo dostop do dela aplikacije za oddajo pooblastila in do dela za poročanje o štipendiranju. Pravice vnosa, spreminjanja in brisanja imajo samo za lastno podjetje/organ (davčna na digitalnem potrdilu) oz. za organe za katere imajo dodeljena pooblastila.

Sklad in Ministrstvo imata dostop do dela aplikacije za poročanje o štipendiranju (vpogled v podatke za vsa podjetja in možnosti vnašanja za podjetja brez certifikatov), administrativnega dela (urejanje določenih šifrantov in parametrov) in dela za potrjevanje pooblastil.

Administrator (izvajalec) ima možnost dostopa do dela aplikacije za kreiranje novih uporabnikov in dela za poročanje o štipendiranju (samo vpogled), ter administrativnega dela (vpogled in spreminjanje podatkov).

Nivoji dostopa posameznega uporabnika se urejajo v šifrantu uporabnikov. Pravico urejanja nivoja dostopa imajo samo uporabniki z nivojem dostopa 00 (administratorji).

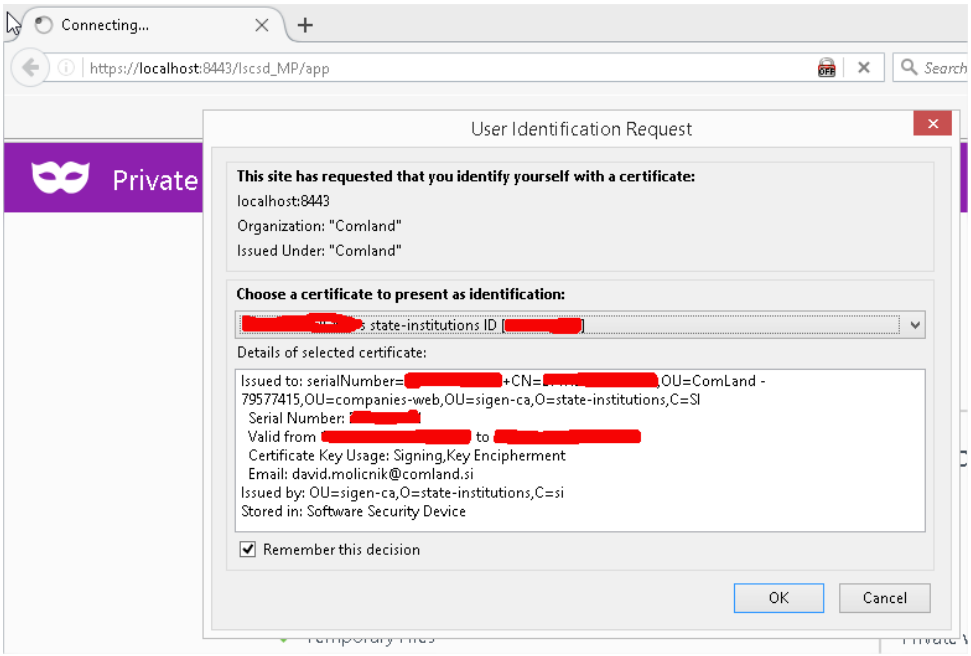
6.1.3 Način prijave v aplikacijo in evidenca prijav

Prijave v aplikacijo se izvajajo preko digitalnih potrdil in potekajo na naslednji način:

1. uporabnik sproži odpiranje prijavne strani v aplikacijo,
2. izbere digitalno potrdilo (Slika 6-3),
3. sproži se postopek preverjanja potrdila; če je preverjanje uspešno, potem se dovoli dostop v aplikacijo; če ni, se o tem obvesti uporabnika in se mu onemogoči dostop do aplikacije (oz. ga preusmerimo na stran sklada).

Za preverjanje veljavnosti potrdila je uporabljena spletna storitev na naslovu:

https://ws.halcom.si/CertificateStatus/CertificateStatus.wsdl



Slika 6-3: Prijava v aplikacijo.

Aplikacija implementira odjemalca za spletno storitev, ki za podano serijsko številko potrdila preverja, ali je potrdilo veljavno oz. je uradno izdano s strani izdajatelja v Republiki Sloveniji.

Vsi poskusi prijav v aplikacijo (uspešni ali neuspešni) se zapišejo v tabelo MP_LOG_PRIJAVA, kjer je zabeležen datum in čas prijave, ter ime, priimek in serijska številka digitalnega potrdila (za neregistrirane uporabnike ali nepravilna potrdila) ali ID uporabnika (za registrirane uporabnike, ki obstajajo v šifrantu uporabnikov).

6.1.4 Postopek kreiranja novega uporabnika in organa

Postopek kreiranja novega uporabnika poteka samodejno po prvi uspešni prijavi v aplikacijo. Programska logika iz potrdila pridobi serijsko številko potrdila. Na podlagi številke potrdila pridobi podatke o lastniku potrdila (ime in priimek osebe, naziv organa in davčno številko organa). Za to obstaja spletna storitev na naslovu:

https://ws.halcom.si/CertificateInfo/CertificateInfo.wsdl

Če iz spletne storitve ni mogoče dobiti davčno številko organa se odpre obrazec za ročni vnos davčne številke. Po vnosu davčne številke se sproži preverjanje ujemanja davčne številke in lastnika potrdila pri spletni storitvi na naslovu:

https://ws.halcom.si/CertificateTaxNumbers/CertificateTaxNumbers.wsdl

Nato se preveri, ali v šifrantu uporabnikov že obstaja uporabnik z določeno številko potrdila. Če ne obstaja se kreira nov zapis / nov uporabnik. Privzet nivo dostopa novega uporabnika je 60.

Preveri se tudi ali v šifrantu organov že obstaja organ z vneseno davčno številko. Če ne obstaja, se pridobi naziv iz šifranta podjetja (CSI_PODJETJA) in se kreira nov zapis v šifrantu / nov organ.

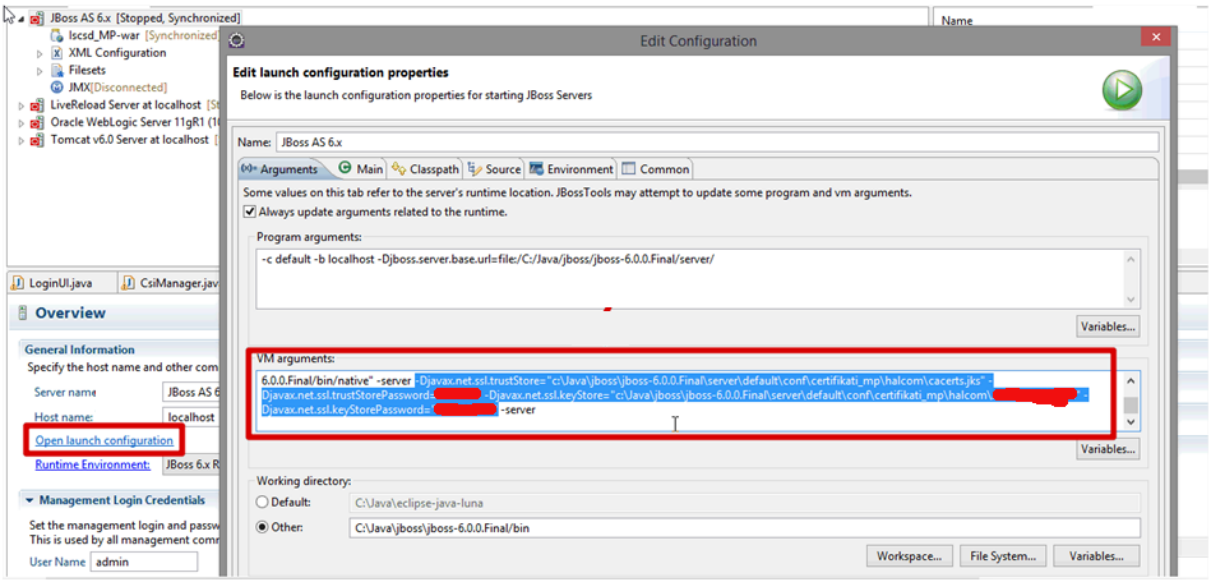
6.1.5 Izbira organa, za katerega se poroča

Po uspešni prijavi v aplikacijo programska logika ugotovi, kateri organ je lastnik digitalnega potrdila. Potem se preveri, ali je organ lastnik potrdila pooblaščen, da v imenu drugega organa ureja podatke o poročanju. Če je pooblaščen se prikaže seznam organov, kjer uporabnik izbere organ, za katerega želi urejati podatke. Na seznamu organov se prikaže organ, ki je lastnik potrdila in poleg njega vsi ostali organi, za katere ima pooblastilo (podatki iz šifranta MP_ORGANI_POOBLASTILA).

V aplikaciji je omogočena zamenjava izbranega organa, za katerega se trenutno ureja poročanje, brez odjave in ponovne prijave v aplikacijo. Opcija se nahaja v glavnem meniju in ponovno prikaže stran za izbiro organa.

6.2 Spremembe aplikacije in razvojnega okolja

Odjemalci oz. spletni brskalniki z aplikacijo komunicirajo prek varne https povezave. V ta namen generiramo pripadajoči hrambi zasebnih (strežniških) ključev *keystore* in javnih ključev *truststor,* ki jim zaupamo (tudi javni ključ od strežnika www.halcom.si). To naredimo s pomočjo orodja *Portacle.* Nato dopolnimo še zagonski ukaz v razvojnem okolju *Eclipse* (Slika 6-4) s parametri Djavax.net.ssl.trustStore, Djavax.net.ssl.trustStorePassword ter Djavax.net.ssl.keyStorePassword in Djavax.net.ssl. keyStore.



Slika 6-4: Sprememba zagonskega ukaza v okolju *Eclipse*.

Nato dopolnimo še nastavitveno datoteko spletne aplikacije *web.xml* z naslednjimi deklaracijami (Slika 6-5):

```
<login-config>
    <auth-method>CLIENT-CERT</auth-method>
</login-config>
<security-constraint>
    <web-resource-collection>
        <web-resource-name>securedapp</web-resource-name>
        <url-pattern>...</url-pattern>
        ...
        ...*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Slika 6-5: Sprememba v *web.xml*.

Dopolnimo še datoteko *server.xml*, kot je označeno na sliki (Slika 6-6).

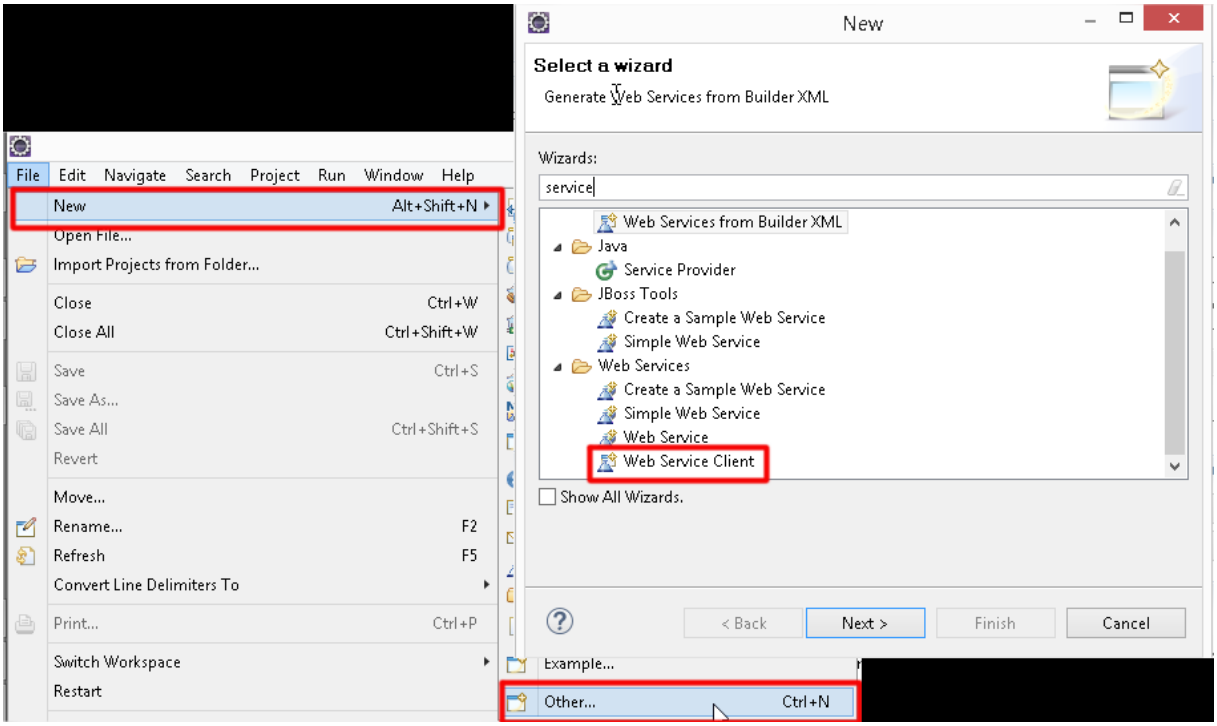
```
<!-- A HTTP/1.1 Connector on port 8080
<Connector protocol="HTTP/1.1" port="${jboss.web.http.port}" address="${jboss.bind.address}"
   redirectPort="${jboss.web.https.port}" /> -->

<Connector protocol="HTTP/1.1" port="${jboss.web.http.port}" address="${jboss.bind.address}" />

<!-- Add this option to the connector to avoid problems with
     .NET clients that don't implement HTTP/1.1 correctly
   restrictedUserAgents="^.*MS Web Services Client Protocol 1.1.4322.*$"
-->

<!-- A AJP 1.3 Connector on port 8009 -->
<Connector protocol="AJP/1.3" port="${jboss.web.ajp.port}" address="${jboss.bind.address}"
   redirectPort="${jboss.web.https.port}" />

<!-- SSL/TLS Connector configuration using the admin devl guide keystore -->
<Connector protocol="HTTP/1.1" SSLEnabled="true"
     port="${jboss.web.https.port}" address="${jboss.bind.address}"
     scheme="https" secure="true" clientAuth="true"
     keystoreFile="[redacted]"
     keystorePass="[redacted]"
     truststoreFile="[redacted]"
     truststorePass="[redacted]"
     sslProtocol = "TLS" />
```

Slika 6-6: Popravek datoteke *server.xml*.

S tem zagotovimo, da teče aplikacija pod *https* protokolom in da spletni brskalnik strežniku pošlje ob prijavi uporabnikov certifikat.

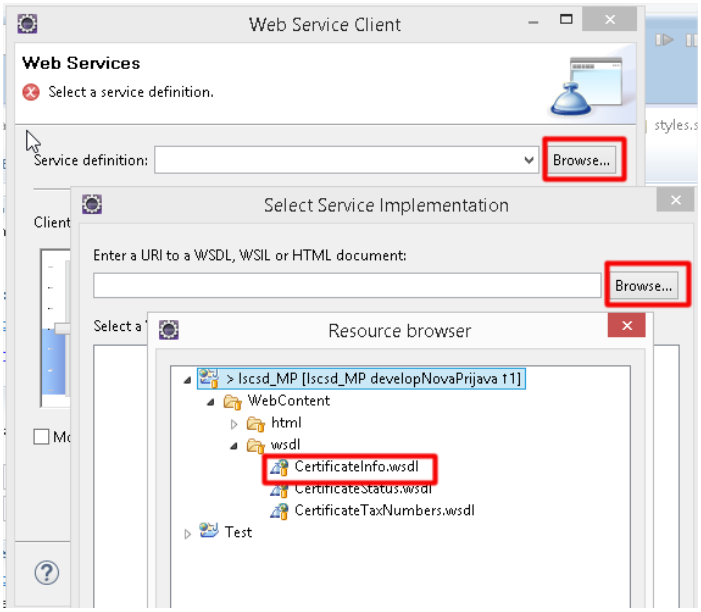
6.2.1 Kreacija spletnega odjemalca

Za implementacijo odjemalca spletnih storitev preverjanje certifikata si iz datotek *CertificateStatus.wsdl„ CertificateInfo.wsdl* in *CertificateTaxNumbers.wsdl* kreiramo t.i. posredniške (*proxy/stub classes*) razrede za dostop do spletnih servisov. To storimo s pomočjo WTP (*Web Tools Platform*) projekta za orodje *Eclipse* (Slika 6-7).



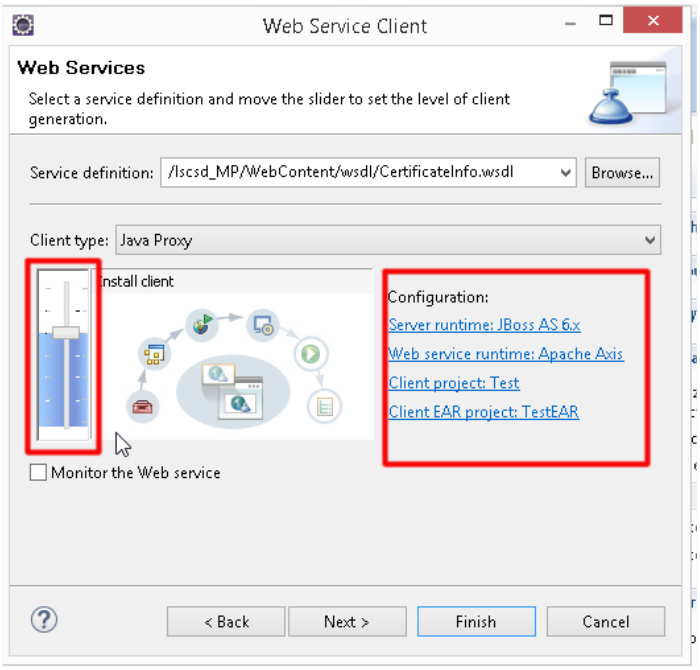
Slika 6-7: Zagon čarovnika za kreacijo spletnega odjemalca.

Ko se nam prikaže čarovnik za kreacijo spletnega klienta si moramo najprej izbrati definicijo spletnega servisa (Slika 6-8).



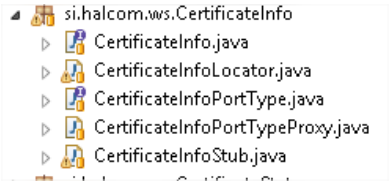
Slika 6-8: Izbira definicije servisa.

Nato si izberemo nivo generacije spletnega klienta, poskrbimo da so nastavitvena polja (*Configuration*) pravilno nastavljena in kliknemo *Finish* (Slika 6-9).



Slika 6-9:Kreacija spletnega klienta.

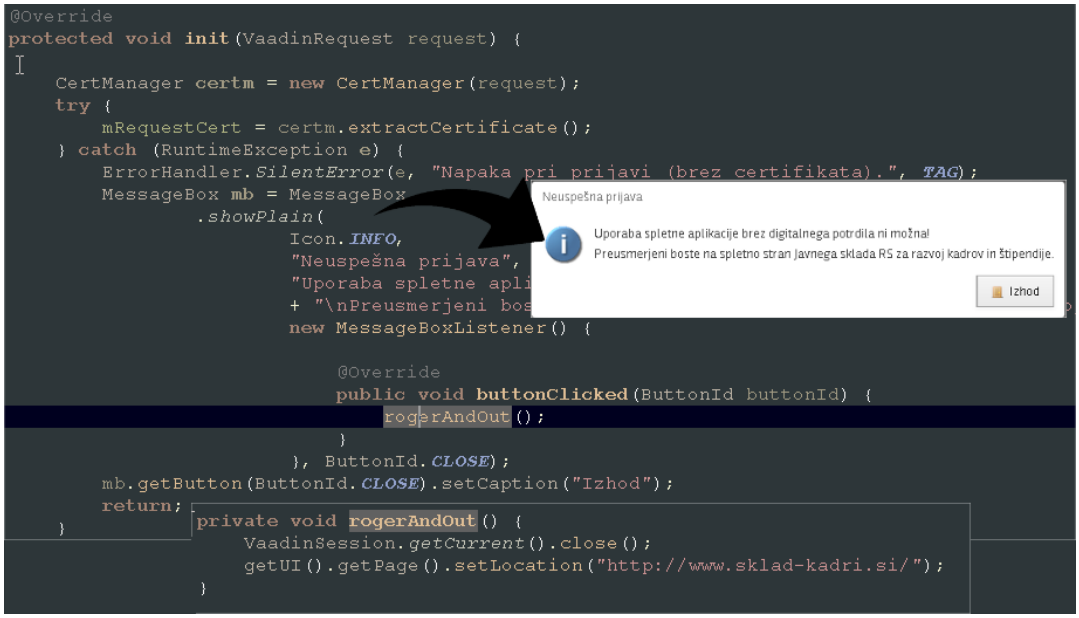
Ko čarovnik zaključi z generacijo odjemalca, se v našem projektu pojavijo naslednji posredniški razredi, ki jih lahko uporabimo v naši aplikaciji (Slika 6-10).



Slika 6-10: Posredniški razredi.

6.2.2 Uporaba spletne storitve v kodi

Vstopna točka v aplikacije je metoda *init(VaadinRequest request)* razširjenega razreda *com.vaadin.ui.UI*. Ob klicu metode najprej pridobimo certifikat iz zahteve odjemalca (Slika 6-11). Če certifikata ne moremo prebrati iz zahteve, se uporabniku prikaže sporočilo in se ga preusmeri na spletno stran Sklada (metoda *regerAndOut*).



Slika 6-11: Vstopna točka – metoda *init()*.

Pri tem si pomagamo z razredom *CertManager* (Slika 6-12), ki si ga izdelamo za pomoč pri delu s certifikati in nam iz zahteve vrne certifikat v objektu tipa *java.security.cert.X509Certificate*.

```
private VaadinRequest mRequest;

public CertManager(VaadinRequest request){
    mRequest = request;
}

public X509Certificate extractCertificate() {

    X509Certificate[] certs = (X509Certificate[]) mRequest.
            getAttribute("javax.servlet.request.X509Certificate");
    if (null != certs && certs.length > 0) {
        return certs[0];
    }
    throw new RuntimeException("No X.509 client certificate found in request");
}
```

Slika 6-12: Razred CertManager.

Ko pridobimo certifikat v objektu tipa *X509Certificate* pokličemo spletno storitev *CertificateStatus.wsdl* prek razreda *HalcomCertStatus* (Slika 6-13), ki si ga predhodno izdelamo. Ta preveri status uporabnikovega certifikata (prijavo) s pomočjo posredniškega razreda *si.halcom.ws.CertificateStatus*, ki smo ga kreirali (6.2.1 Kreacija spletnega odjemalca) iz definicije spletne storitve.

```
public class HalcomCertStatus {
    // web service endpoint
    final private static String ENDPOINT = "https://ws.halcom.si/CertificateStatus";
    // web service output parameters
    private BigIntegerHolder returnCode = new BigIntegerHolder();
    private StringHolder returnText = new StringHolder();
    private BigIntegerHolder certificateStatus = new BigIntegerHolder();
    private CalendarHolder producedAt = new CalendarHolder();
    private CalendarHolder thisUpdate = new CalendarHolder();
    private CalendarHolder nextUpdate = new CalendarHolder();
    private BigIntegerHolder revocationReason = new BigIntegerHolder();
    private CalendarHolder revocationDate = new CalendarHolder();
    private X509Certificate mCertToCheck;
    public HalcomCertStatus(X509Certificate cert) {
        mCertToCheck = cert;
    }
    public void getCertStatus() throws CertificateException, KeyStoreException, NoSuchAlgorithmException, IOException {
        // set web service endpoint address
        CertificateStatusStub stub = new CertificateStatusStub();
        stub._setProperty(javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY, ENDPOINT);
        // web service input parameters
        String actionPolicy = "empty"; // ignored
        StringHolder other = new StringHolder("ditto"); // input/output -
        System.out.println("Verifying certificate status for:");
        System.out.println("[issuer]: " + mCertToCheck.getIssuerDN());
        System.out.println("[subject]: " + mCertToCheck.getSubjectDN());
        System.out.println("actionPolicy: " + actionPolicy);
        System.out.println("other: " + other.value);

        CertificateStatusPortType cs = (CertificateStatusPortType) stub;
        cs.getCertificateStatus(mCertToCheck.getEncoded(), actionPolicy, other, returnCode,
                returnText, certificateStatus, producedAt, thisUpdate, nextUpdate,
                revocationReason, revocationDate
        );

    }

    public BigIntegerHolder getReturnCode() {
    public void setReturnCode(BigIntegerHolder returnCode) {
    public StringHolder getReturnText() {
    public void setReturnText(StringHolder returnText) {
```

Slika 6-13:Razred HalcomCertStatus.

Če pride med klicanjem spletne storitve do napake (ene od java izjem *CertificateException, KeyStoreException, NoSuchAlgorithmException* ali *IOException*), se o tem obvesti uporabnika in preusmeri na spletno stran Sklada. Prijavo se, ne glede na to ali je uspešna ali neuspešna, zabeleži v tabelo MP_LOG_PRIJAVA.

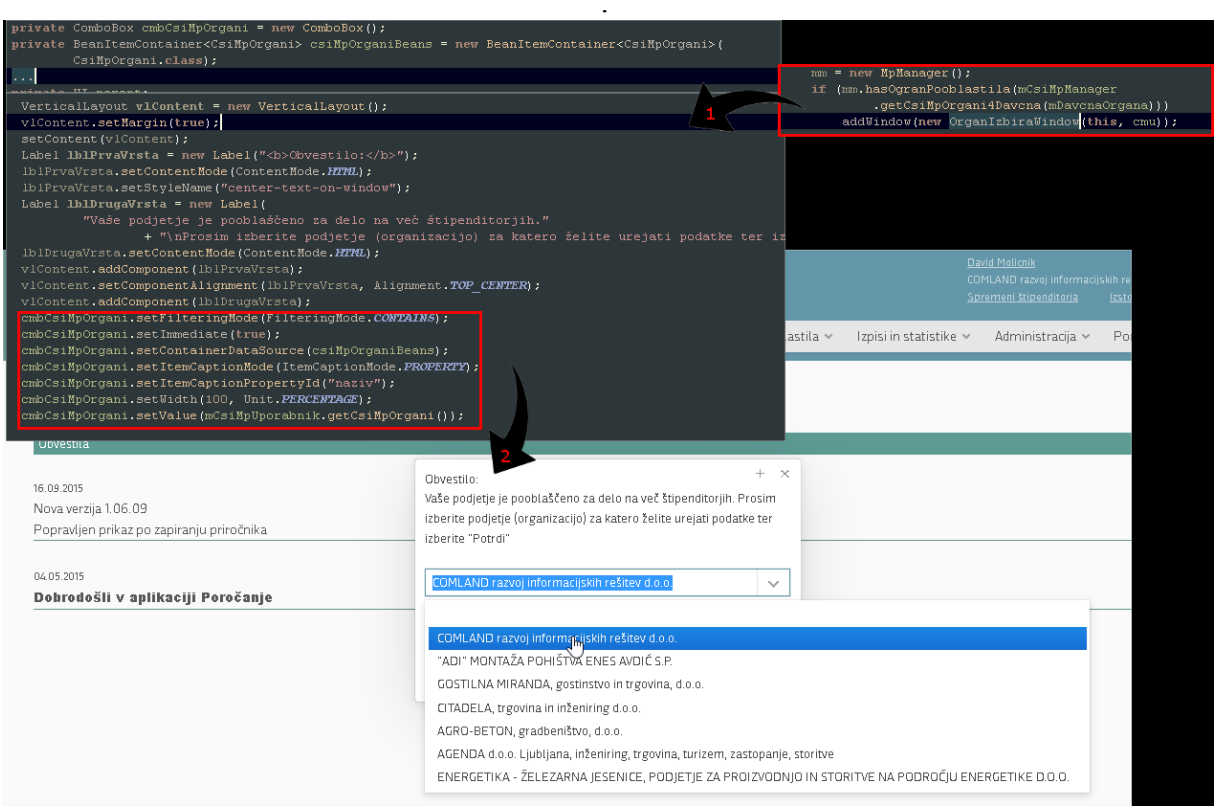
Če spletna storitev vrne status 0 (Slika 6-14), se prijava uporabnika nadaljuje s podobnima klicema spletnih storitev *CertificateInfo.wsdl* in *CertificateTaxNumbers.wsdl* ter izbiro organa, za katerega bo uporabnik urejal podatke (Slika 6-15).

```
try {
    hcs.getCertStatus();
    if (hcs.getCertificateStatus().value.equals(BigInteger.valueOf(0))) {
        mCertOk = true;
    } else {
        mCertOk = false;
    }
```

Slika 6-14: Branje statusa, ki ga vrne spletna storitev.

Za uporabnika se najprej preveri, če je že vnesen in, če ima kakšna pooblastila za poročanje v imenu drugih organov. Ta seznam se s pomočjo JPA klica iz podatkovne zbirke prenese v *BeanItemContainer*, ki je ena od implementacij podatkovnega modela (Slika 4-7) in vsebuje objekte *JavaBean,* ki so zaviti v referenčni tip (wrapper) BeanItem [8]. To pa je implementacija vmesnika Item iz podatkovnega modela ogrodja Vaadin, ki lahko ovije katerikoli preprost java objekt (POJO) [8].

Objekt *cmbCsiMpOrgani* je kombinirano polje tipa *com.vaadin.ui.ComboBox.ComboBox()* . To je vizualni gradnik s katerim komunicira uporabnik pri izbiri podjetja / organa. Podatkovni vir tega objekta pa je prej omenjeni *BeanItemContaine.*



Slika 6-15: Izbira podjetja / organa za poročanje.

7. Sklepne ugotovitve

Ogrodje *Vaadin* je zelo dobro načrtovano in implementirano ogrodje s katerim lahko začnemo hitro razvijati aplikacijo. Bogato spletno Aplikacijo lahko v celoti razvijemo na strežniški strani, brez ene same vrstice kode na odjemalčevi strani. Tako se dejansko lahko v celoti posvetimo razvoju poslovne logike. Razvoj je zelo podoben razvoju namiznih aplikacij, zato lahko razvijamo spletne aplikacije brez podrobnega predhodnega znanja o spletnih tehnologijah: JS, HTML, AJAX itd. Seveda pa imamo odprta vrata za razvoja svojih lastnih UI komponent.

Pomembno je dejstvo, da skupnost iz dneva v dan narašča, kar doprinese še k večji popularnosti in prepoznavnosti ogrodja. Hkrati pa se povečuje število razvijalcev, ki razvijajo nove komponente za to ogrodje.

Zelo pomembno je tudi dejstvo, da lahko v razvoj spletne aplikacije takoj vključimo razvijalce, ki so bolj vešči programiranja GUI vmesnikov in ne toliko spletnih aplikacij (torej so lahko popolni »*Java* monogloti«, lahko so tudi npr. *C#* razvijalci).

Ker je poslovna logika ločena od prikaza in podatkovne plati, se lahko delo na projektu lepo porazdeli med več skupin in s tem še dodatno pohitrimo razvoj.

Literatura

[1] 10 criteria for choosing the correct framework (2016) Dostop na: http://symfony.com/ten-criteria

[2] *Nav*odila_za_Distribucijski_modul (2016) Dostop na: http://arhiv2014.skupnostobcin.si/

[3] OPIS OBSTOJEČEGA SISTEMA ISCSD S POPISOM APLIKACIJ IN STREŽNIKOV (2016) Dostop na: http://www.mddsz.gov.si/fileadmin/mddsz.gov.si/pageuploads/dokumenti__pdf/word/jr_dok_gradivo_csd_jan11_iscsd.doc

[4] What are the benefits of using Vaadin over plain GWT (2016) Dostop na: https://www.quora.com/What-are-the-benefits-of-using-Vaadin-over-plain-GWT

[5] What is GWT? (2016) Dostop na: http://www.tutorialspoint.com/gwt/index.htm

[6] Google Web Toolkit (2016) Dostop na: https://en.wikipedia.org/wiki/Google_Web_Toolkit#Features

[7] Vaadin (2016) Dostop na: https://en.wikipedia.org/wiki/Vaadin#Features

[8] Book of Vaadin: Vaadin 7 Edition - 7th Revision, Marko Grönroos (2016) Dostop na: https://**vaadin**.com/**book**

[9] Informacijska varnost (2016) Dostop na: https://sl.wikipedia.org/wiki/Informacijska_varnost